

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-153640

(43)Date of publication of application : 06.06.2000

(51)Int.Cl.

B41J 2/52

H04N 1/23

(21)Application number : 11-327652

(71)Applicant : XEROX CORP

(22)Date of filing : 18.11.1999

(72)Inventor : CURRY DOUGLAS N

(30)Priority

Priority number : 98 195165

Priority date : 18.11.1998

Priority country : US

(54) PRINT METHOD AND PRINTER CONTROLLER

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a printer controller and a method therefor which use print data and printer control commands so that a laser power to be used for forming objects of each type to page images defined with use of a PDL or the like is controlled or selected to be optimum.

SOLUTION: The print method includes a step of distinguishing a byte space of image data including data of different types, a step of printing byte image data of a first type with use of a laser power drive signal at a first step point, and a step of printing byte image data of a second type with use of the laser power drive signal at a second step point. The first step point is different from the second step point.

LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

*** NOTICES ***

Japan Patent Office is not responsible for any damages caused by the use of this translation.

- 1.This document has been translated by computer. So the translation may not reflect the original precisely.
- 2.**** shows the word which can not be translated.
- 3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]

[Claim 1] The print method that have a step which distinguishes between cutting tools of image data containing data of a different type, a step which prints cutting tool image data of the 1st type on the 1st step point using a laser power driving signal, and the step which prints image data of a cutting tool of the 2nd type on the 2nd step point using said laser power driving signal, and said 1st step point differs from said 2nd step point.

[Claim 2] It is the printer controller which adjusts a driving signal set point used in order to print the cutting tool's image data based on a type of an object corresponding to a cutting tool's image data. It has an image printing terminal which receives print data and meta-bit data. It has an image processing system containing an object optimization image formation modulation subsystem with which said image printing terminal is selectively controlled by meta-bit data transmitted to said image printing terminal. Said meta-bit controls said object optimization image formation modulation subsystem to choose between driving signal set points which change with types of an object with which said cutting tool's image corresponds. A printer controller which affects said formation of said object corresponding to image data said whose driving signal set point is said cutting tool.

[Claim 3] A printer controller according to claim 2 which is built into an image formation device and chosen from a group to whom said image formation device changes from a facsimile machine, a printer, a digital process copying machine, and a raster output scanner.

[Translation done.]

*** NOTICES ***

Japan Patent Office is not responsible for any damages caused by the use of this translation.

- 1.This document has been translated by computer. So the translation may not reflect the original precisely.
- 2.**** shows the word which can not be translated.
- 3.In the drawings, any words are not translated.

DETAILED DESCRIPTION

[Detailed Description of the Invention]

[0001]

[The technical field to which invention belongs] This invention relates to the method and equipment for adjusting max, saturation laser, or LED power in a printing equipment. This invention relates to the method for controlling exposure level at details based on the type of an image object by which current exposure is carried out more.

[0002]

[Description of the Prior Art] The creator (implementer) of the page image for which it had been depending on the GURAFIKU artist for some time in order to compound and print a page image with development of a digital color workstation, a copying machine, and a printer can use instead the computer connected to the digital color copying machine / printer, and can generate, compound and print a page image by itself.

[0003] However, in order to generate a page image and to disassemble a page image into a print engine instruction, the digital system of such conventional technology for controlling a print engine was single, and was processing the page image as an image of one. Thus, in the page image optimized for a text / line art, when a RF halftone screen is used, the text portion of the page image is very sharp. However, the fixed color portion of a page image includes clear spots-ization from a printer noise. Furthermore, the color portion by which the page image was sampled, and a sweep (sweep) portion include clear outline formation by lack of sufficient gray level which may be used on a RF screen.

[0004] In the page image optimized for the big fixed color portion, the halftone screen designed especially in order to hide the instability of a printer generates the fixed color field which is quality, does not have a text and does not have the artifact. However, the sharpness of a text falls, the gray value for every tint is seldom related, and the color portion and sweep portion which were sampled cannot be permitted. Since each dot level is independently designed not related with other level, the step of the gray level is not smoothly carried out from one level to the following level.

[0005] In the page image optimized to the color portion and sweep portion which were sampled, since a low frequency halftone screen is used by the gray level which can be used, a sweep portion and the sampled color portion show higher quality. However, a text is expressed with low quality and a fixed color portion shows clear pattern-ization.

[0006] However, in the conventional system which processed each image as a single bitmapped image or a cutting tool map image, optimizing an image to all one types of object needed to compromise with the image quality of the object of other types. therefore , the digital color copying machine / printer , and the method of maintain the advantage which can be use when generate a page image using a microcomputer , and effectiveness be required of the conventional technology at the same time it output to the print engine which generate a page image , decompose and can optimize the print property of the object according to individual similarly in the synthetic GURAFIKU art field by handicraft .

[0007] GrAphicAl which is used with a Page Description Language [like PostScriptTM] (PDL) such whose a page image is, InterpressTM, and WindowsTM DisPlAy InterfAces (GDI) and

Hewlett-PACKard Printer CommAnd LAnguage (PCL-5) etc. -- it is used and generated.

[0008] It is known that the laser beam printer product marketed will expose "monochrome" image using the powerful laser power which is not common. Enhancement of the contrast between a white image and a black image is benefited for this, and the enclosure field by this fault exposure is liked aesthetic for those who see. All the image ranges are provided with this enhancement laser power.

[0009] However, this reinforced contrast makes it impossible to generate a thin line in connection with it. Furthermore, although fault exposure of a text and/or a line art is liked aesthetic, since the reinforced laser power carries out fault exposure of the halftone cell, fault exposure of a halftone image is disadvantageous. Carrying out fault exposure of the halftone cell decreases the control covering the whole dynamic range. Since enhancement of laser power covers all the dynamic ranges of a print engine and influences transition of a form function, this is produced. If laser power is increased especially, a form function is saturated too much quickly and cannot change in the high edge of a dynamic range.

[0010]

[Problem(s) to be Solved by the Invention] This invention makes it a technical problem to offer the printer controller equipment and the method of using print data and printer control command so that the laser power used in order to form the object of each type in the page image defined using PDL etc. may be controlled or chosen the optimal. Print data and printer control command may be changed from the page image defined using PDL, and may be offered according to other well-known mechanisms in the conventional technology.

[0011] This invention generates the "meta-bit" information which includes laser power information further based on the object type of the various objects which form a page image, i.e., the information about the method of improving each cutting tool's image data most a rendering, and offers the object optimization printer control unit and method of sending these meta-bit data to an image printing terminal (IOT).

[0012] This invention offers the object optimization printer control unit and method of determining laser power automatically further based on the object type to the object of each independence of a page image.

[0013]

[Means for Solving the Problem] A page image which is described as an example of a step of equipment used and a method with this invention using PDL, and is stored as a series of PDL commands is inputted into an object optimization electro nick subsystem (OOESS). A PDL decomposition means disassembles a page image described by PDL to a data structure showing a page image. In this data structure, information relevant to an independent image object is held. This information includes information relevant to a property according to individual of others, such as a type of an object and the maximum laser modulation set point, a color, the optimal color space, and layer (layer) information.

[0014] Once it changes a page image with which a PDL decomposition means was described by PDL, an image printing terminal (IOT) and a command instruction generating means will change a data structure into a series of command instructions, color assignment, and a meta-bit rendering instruction for every scan line of a page image at a copying machine/printer, and a general twist target. In order to determine a type of each object, you may be automatically generated by meta-bit generating means to analyze each object, and a meta-bit may be clearly set up by page image creator between generation of PDL description of a page image. If a command instruction, color assignment, and a meta-bit are generated for every scan line of a page image, they will be outputted to an IOT controller.

[0015] An IOT controller receives a command instruction, color assignment, and a meta-bit which were generated between decomposition processings. An IOT controller combines a fixed color and sampled color data, and transmits it to IOT with meta-bit data.

[0016] In a gestalt of implementation of instantiation of the 1st of IOT, IOT can include two or more halftone screen generators, a threshold circuit, a color space conversion circuit, and a laser power section circuit. Cutting tool width-of-face color data and a meta-bit which are outputted from an IOT controller are inputted into IOT. A meta-bit is used in order to determine

a threshold circuit [which halftone generator or] to be used to color data, and which color conversion is applied by color space conversion circuit. Moreover, a meta-bit is used in order to choose an alternative laser power set point which should be used in order to print image data, i.e., the maximum laser reinforcement. If IOT generates raster data from color data and a meta-bit, raster data containing laser data on the strength will once be outputted to a marking subsystem, in order to form a page image on an output sheet.

[0017] Or sample color-data compression / compression discharge circuit, mask-data compression / compression discharge circuit, fixed color compression / compression discharge circuit, and a command data compression / compression discharge circuit may be included in an IOT controller of a bus interface and integrated KOMBANA (coupler). Therefore, as for these compression/compression discharge circuits, data of various types of a data structure is compressible the optimal for a transfer to an IOT controller on a system bus. Furthermore, in a gestalt of implementation of instantiation of the 1st of an IOT controller, a color space converter is included in an IOT controller. By including a color space converter in an IOT controller, color space conversion to an object from which a page image differs can be exactly performed before a print, and may be further optimized based on an object type. Furthermore, including a color space converter in an IOT controller extends the range of IOT which may be used with a system of this invention. Or other gestalten over these compression/compression discharge circuits may be used.

[0018] Furthermore, since color data outputted to IOT are generated, two color space converters may be used. Thus, 4 bytes of data showing each of four color layer separation, and C, Y, M and K may be outputted simultaneously. By outputting simultaneously color data to all 4 color layer separation, a high page output is attained [rather than] to IOT, and the larger range of IOT may be used with a system of this invention.

[0019] A step which distinguishes between cutting tools of image data which contains data of a type with which the 1st modes differ as a concrete mode of this invention, A step which prints cutting tool image data of the 1st type on the 1st step point using a laser power driving signal, It is the print method that have a step which prints image data of a cutting tool of the 2nd type on the 2nd step point using said laser power driving signal, and said 1st step point differs from said 2nd step point.

[0020] It is the printer controller which adjusts a driving signal set point used in order that the 2nd mode may print the cutting tool's image data based on a type of an object corresponding to a cutting tool's image data. It has an image printing terminal which receives print data and meta-bit data. It has an image processing system containing an object optimization image formation modulation subsystem with which said image printing terminal is selectively controlled by meta-bit data transmitted to said image printing terminal. Said meta-bit controls said object optimization image formation modulation subsystem to choose between driving signal set points which change with types of an object with which said cutting tool's image corresponds. Said driving signal set point is the printer controller which affects said formation of said object corresponding to image data which is said cutting tool.

[0021] In the 2nd mode, a printer controller is built into an image formation device, and the 3rd mode is chosen from a group to whom said image formation device changes from a facsimile machine, a printer, a digital process copying machine, and a raster output scanner.

[0022]

[Embodiment of the Invention] As mentioned above, unless a creator (generation person) covers the whole page image and inserts a printer dependence halftone clearly, the conventional processing for disassembling the page image described by the Page Description Language (PDL) needs to reach a compromise, when printing the bit map or cutting tool map generated by disassembling the page image described using PDL. The following description may be used with the object optimization rendering system and the method that all above-mentioned systems equivalent to PDL are described below, although referred to only to PDL.

[0023] In the conventional system for disassembling and printing the page image described using PDL as mentioned above, namely, the various objects which consist of a page image It is changed into a device coordinate from a PDL coordinate, and a stroke is thickly made thick. An

object is changed into a series of boxes corresponding to the location of the object on a scan line, respectively, and these boxes are loaded to a bit map, when it is a cutting tool map (or an image black/white). By writing the box formed by decomposing the object of a page image in a cutting tool map, all distinction during the type with which objects differ is lost.

[0024] With this, the page image reversely described by PDL in this invention is disassembled so that distinction between the object types which consist of a page image may be maintained. By maintaining the object type of the object from which it differs in a page image, processing of a different object type can be optimized to the type of an object.

[0025] Before, in order to form the text image of a sharp edge at the same time coarse screen halftone generating processing is used, in order to form the background of concentration of changing continuously, detailed halftone generating processing was used. The optical density which changes continuously [a background] may be maintained by using optimal halftone generating processing for every type of an object. Simultaneously, the optimal halftone generating processing for maintaining the sharp edge of a text image may be maintained.

[0026] Thus, while being used in order to form the text image with which the RF halftone screen was optimized, it may be used in order that a hue (tint) assignment halftone set may form the block with which the fixed color was optimized, and it may be used in order that a low frequency halftone screen may form further the color picture and sweep (sweep) which were optimized and sampled.

[0027] By maintaining the object type of the various objects which consist of a page image, other features of an object may fully be optimized like color space conversion of the object to the cyanogen (C) used by IOT from the color space used by the Page Description Language, a Magenta (M), yellow (Y), and (Black B) color space. Each step of color transform processing may actually be optimized like undershirt color clearance processing and tone (gradation) playback curvilinear processing. Similarly, compression processing can be optimized by the object and, thereby, makes min the transition load over the storage resource (resource) and the disassembled page image consumed.

[0028] Furthermore, depending on the degree of the optimization demanded, the level from which distinction of an object type differs may be offered. That is, probably, in some instances (example), it will be enough to direct that an object is colored in a fixed color and to optimize a halftone screen and all processing functions like color space conversion on this level. In other instances, in order to specify one of the halftone screen frequency of many of color space many of [possible / possible one or], or the angles, the level of additional distinction may be required and those all are suitable for a fixed color object. [many of]

[0029] Furthermore, a processing function is optimized in order to offer the suitable level of the laser beam exposure used in order to form each object in an image. In order to form the outstanding halftone dot, a form function and the whole dynamic range of the optical exposure to a shadow (shading) must continue and cohere through a mid tone from highlights. In halftone formation, the problem which originates in fault exposure of a halftone dot like the case where fault exposure of a text and/or the line art is carried out will not be left with an unexposed field not exposed in the range, if a form function moves to the mid tone portion of a dynamic range. Therefore, the halftone acquired serves as perfect overlap, and a form function goes into a shadow field, when [of all dynamic ranges] it is still on the way. Therefore, since the form function has already changed to the place of a dynamic range which is in the high end of a dynamic range from a certain place low-end, the high end of all dynamic ranges does not influence the print quality of an image.

[0030] However, the appearance of a text and a line art is more desirable when fault exposure of a text and/or the line art field is carried out. A reason is that contrast will be emphasized if a text and/or a line art field are exposed by higher power. Although fault exposure looks better, the precision in the rendering of a very thin line will be lost in all images. However, the lost precision is not large as compared with the appearance by which the text and/or the line art were improved. Therefore, it is advantageous to make it larger than the laser power between processings of the laser power between processings of a text and/or a line art of a halftone dot.

[0031] These differences between a text and line art generation, and halftone generation are

based on size and the problem of control. The size of the feature according to individual of the object in a text or a line art object is larger than the feature according to individual in a halftone object. Therefore, forming a text or a line art generates an image coarser than the image generated by half toning. If fault exposure of a text or the line art is carried out, since it will be estranged without the feature according to individual seldom approaching, as for fault exposure, the time of fault exposure of the halftone dot being carried out and ** do not fall a form function to whenever. Furthermore, forming a text and a line art does not need control of a tone as well as halftone formation. In half toning, a form function changes from a circle configuration to a diamond configuration, and it returns to a circle configuration as the image concentration of a halftone dot changes from highlights concentration to shadow concentration through mid tone concentration along with all the dynamic ranges of a print engine. In a full color image, since exact control of the field covered by these form functions controls the color perceived by those whom the ratio of the field covered by a Magenta, yellow, and cyanogen looks at, it is important.

[0032] The object optimization electronic subsystem (OOESS) 100 for changing into the raster data which can be used with the image printing terminal (IOT) 170 to the PDL form of image data, as shown in drawing 1 is Sun of a desirable general purpose computer like a personal computer, California, and Maung Teng. Microsystems It is provided by engineering workstation like SunSpArcTM by the shrine manufactured, a minicomputer, etc. OOESS100 can include an internal PDL file source means like the program for generating the PDL expression of a page image like a program which generates PostScriptTM and an InterPressTM compatible document, the program which generates the GDI expression of a page image, and the program which generates the graphical commands set expression of a ** page image. One type of a graphical commands set is Hewlett-PAckArd used in order to operate a laser beam printer and/or an ink jet printer. It is the commands set of PCL-5. Therefore, it should be understood that term "PDL" should be interpreted as including all the types that describe a page image of expression instead of generating the bit / cutting tool map of a page image.

[0033] Or it can be generated, and the PDL expression of a page image can be rather received from a certain remote PDL file source means 112 like the general purpose computer of a remote place connected to OOESS100 through nonvolatile memory, a local network, or a modem rather than it is decomposed directly. Therefore, it should be understood that the PDL file showing a page image can obtain from all the conventional sources.

[0034] Once the PDL file showing a page image is inputted into OOESS100, it will be transmitted to memory 150 through a bus 114. Next, a PDL file is decomposed by the PDL decomposition means 130. The PDL decomposition means 130 reads the PDL file stored in memory 150, and forms the data structure which decomposes it and is shown in drawing 28 thru/or drawing 31. The data structure shown in drawing 29 contains the rendering tag which directs an object list and an object type. Actuation of a PDL decomposition means is stated to details below.

[0035] If the PDL decomposition means 130 generates a data structure in the memory 150 which stores the various objects of the page image generated from a PDL expression, reading appearance of the data structure stored in memory 150 will be carried out by the command instruction generating means 140. The command instruction generating means 140 changes into a series of command instructions corresponding to color data, bit mask data, and meta-bit data the data structure stored in memory 150 for every scan line. These command instructions, color data, bit mask data, and meta-bit data are stored in memory 150 as shown in drawing 31.

[0036] As shown in drawing 2, it is foreknown that the IOT controller 160 is connected to a bus 114 through a bus interface 1610. A SunSPArcTM workstation is used in the gestalt of desirable operation of OOESS100. Thus, a bus 114 is SBus (S bus), and a bus interface 1610 may be designed so that it may work with SBus114. However, a bus interface 1610 may be designed so that it may work with the specific bus 114 which may be used also in any, such as the conventional personal computer, an en JIRIA ring workstation like SunSpArcTM, and a microcomputer.

[0037] As shown in drawing 2, a bus interface 1610 offers 32-bit connection to a bus 114. Thus, a bus interface 1610 can input 4-byte WORD between each clock cycle. In the gestalt of desirable operation, a bus interface 1610 can be read by 64-byte burst by reading 16 4-byte

WORD continuously by the continuous clock cycle. Furthermore, the bus interface 1610 in the gestalt of desirable operation offers the direct memory address (DMA) to memory 150 through a bus 114.

[0038] If a bus interface 1610 receives the data of a 4-byte portion from memory 150, data will be distributed to one set of the five-set FIFO (FIFO) data register. These five FIFO contains the sample color channel FIFO 1620, the mask-data channel FIFO 1622, the fixed color channel FIFO 1624, the meta-bit channel FIFO 1626, and the command instruction channel FIFO 1628.

[0039] As shown in drawing 2, the sample color channel FIFO 1620 consists of two banks of FIFO, and each bank of FIFO consists of two 512 address X 9-bit width of face FIFO. Thus, each FIFO of each bank of FIFO receives 1 byte of the 4-byte WORD received with the bus interface 1610. Furthermore, a bus interface 1610 generates the data of the bit of four additions, and one of the data of the bit of this addition is stored in each of four FIFO as 9th bit. Each of the bit of these excesses is used in order to carry out the flag of whether the data of the cutting tool relevant to it is an effective data, or it is an invalid data (a flag is set). Since the ejection of DMA data is made only in the word boundary, it sometimes happens that the actual data to the sampled image starts within WORD. In this case, a flag is set as an invalid, the cutting tool in the WORD preceded with initiation of actual data is not printed, and these cutting tools may be canceled.

[0040] The mask-data channel FIFO 1622 consists of a single 256 address X 16-bit width-of-face FIFO data register. The fixed color channel FIFO 1624 consists of one bank of two FIFO, and each FIFO of a bank consists of one 256 address X 9-bit width-of-face FIFO data register. Since each of a mask FIFO 1622 and a color FIFO 1624 can store 2 bytes per write cycle, data can be continuously offered to both a mask FIFO 1622 and the fixed color FIFO 1624 using single [4 bytes of] transfer of the data inputted into a bus interface 1610. Since the only 16-bit width-of-face internal bus connects a bus interface 1610 to a mask FIFO 1622, the fixed color FIFO 1624, and a command FIFO 1628 and the only 8-bit bus connects a bus interface 1610 to the meta-bit FIFO 1626, it is required to be used in order that two clock cycles may write one 4-byte imprint in a mask FIFO 1622, a color FIFO 1624, and a command FIFO 1628, and for four clock cycles to write a 4-byte imprint in the meta-bit FIFO 1626.

[0041] The meta-bit FIFO 1626 consists of a single 512 address X 8-bit width-of-face FIFO data register. The command instruction FIFO 1628 consists of a single bank of the 512 address X 8-bit width-of-face FIFO data register of a pair.

[0042] Each output of FIFO 1620 thru/or 1628 is connected to multi-channel KOMBANA (coupler) 1630. The output from a mask FIFO 1622 is made serial. Thus, the only single bit connection is offered from a mask FIFO 1622 to the multi-channel combiner 1630. Similarly, the sample color FIFO 1620 and the fixed color FIFO 1624 are multiplexed by (2-to-1) from 2 to 1. Thus, the only 18-line internal connection is offered to the sample color FIFO 1620 to a multi-channel combiner. Each bank of a pair of 9-bit width of face FIFO outputs the data by turns. Similarly, only 9-bit width-of-face connection is offered between the fixed color FIFO 1624 and the multi-channel combiner 1630. The 9-bit each width of face FIFO of the fixed color FIFO 1624 outputs the data to a multi-channel combiner by turns.

[0043] As the meta-bit FIFO 1626 and the command instruction FIFO 1628 are indicated by contrast to be FIFO 1620 thru/or 1624 to drawing 2, the full width-of-face connection with the multi-combiner 1630 is offered, respectively.

[0044] The multi-channel combiner 1630 fully explained by the following in relation to drawing 3 combines the data from the sample color channel 1620 and the fixed color channel 1624 to the output of the single stream to data FIFO 1642 based on the bit map data received from the instruction received from the command instruction FIFO 1628, and the mask FIFO 1622.

[0045] Data FIFO 1642 consists of a 4K address X 9-bit width-of-face FIFO data register. Since a multi-channel combiner outputs data to data FIFO 1642 from a 8-bit width-of-face cutting tool, the 9th bit of data FIFO 1642 is used for storing the 1st bit (beginning) from the meta-bit FIFO 1626. Moreover, the multi-channel combiner 1630 is connected to the output meta-bit FIFO 1640. The output meta-bit FIFO 1640 consists of a 4K address X triplet width-of-face FIFO data register. The multi-channel combiner 1630 divides a 8-bit each meta-bit cutting tool

to at least two 4-bit width-of-face meta-bit nibbles. If the 1st bit of a 4-bit each nibble is outputted to data FIFO 1642, the 2nd to the 4th bit of a 4-bit each nibble will be outputted to the output meta-bit FIFO 1640. The number of the meta-bits outputted from a multi-channel combiner may actually be 1, 2, 4, or 8 depending on the number of the types of the object which should be distinguished, and the number of the level of each type of an object which should be performed.

[0046] Each command instruction defines the adjustable amount of the scan line only in relation to a single scan line so that it may be stated below. Therefore, since the output of a multi-channel combiner cannot be appropriately synchronized to the page and line of IOT, concurrency input port 1650 receives a page synchronous (sync) signal, a line synchronizing signal, and an IOT clock signal, and inputs into the multi-channel combiner 1638.

[0047] An output 1640 and FIFO 1642 is connected to the concurrency output port drivers 1652 and 1654, respectively. The output from these concurrency ports 1652 and 1654 is connected to the input port of IOT170.

[0048] Being received by IOT170 as data with which data FIFO 1642 should be processed for a print should be understood. However, this invention offers the print data of each cutting tool from data FIFO 1642 synchronizing by 1, 2, 4, or 8 meta-bit from the output meta-bit FIFO 1640, and being accompanied by it further. These meta-bits specify the processing which should be made to each data byte, before being printed. As explained, this optimal processing changes with each data byte depending on the object from which each data byte was extracted. As shown in drawing 34, in the gestalt of suitable operation of the IOT controller 160, IOT170 has the image processing system 1710 containing the magnitude of the subsystem which is selectively controlled by the meta-bit data transmitted to IOT170 and by which object optimization was carried out. Thus, a meta-bit channel is made to process so that close may differ to each cutting tool's print data depending on the coming object.

[0049] For example, a meta-bit can be chosen among many halftone generators 1712. One halftone generator may be used with the tagged data byte, when it comes from an object like the text of the color which should use the frequency halftone generator between altitude. Other objects may be low spatial-frequency high color resolving halftone generators used with the data byte obtained from a photograph, pictures, etc. A meta-bit select is [to halftone generating like control of the halftone screen angle for every object rather than] possible for detailed control. In addition to selection of a halftone, deal with the problem resulting from having a dialog mutually with those edges of various objects. The object optimization color conversion subsystem 1714, the object optimization tone playback curvilinear modification subsystem 1716, the object optimization spatial filter subsystem 1718, and the object optimization trapping subsystem 1720, And the object optimization engine noise control subsystem which is going to control the problem of the various print systems which appear so that it may differ in a different object, In order to generate the adjustable result for which it depends on a list at an object like other object optimization image-processing subsystems, in an image processing system 1710, the image-processing subsystem of many additions may be controlled by the meta-bit.

[0050] IOT170 contains the laser power selected subsystem 1724 in an image processing system 1710. A meta-bit controls the laser selected subsystem 1724 of an image processing system 1710, and chooses it between different set points depending on the meta-bit relevant to each specific cutting tool of image data. Especially the object optimization laser power selected subsystem 1724 changes the set point of the laser power used in order to carry out the rendering of the cutting tool's image data based on whether the cutting tool of image data is in a halftone object, a text, or a line art object. In order to carry out the rendering of the object, the object optimization laser power selected subsystem 1724 enables selection between two or more set points so that the most suitable laser power may be offered.

[0051] For example, when directing that the cutting tool's image data is the portion of a text or a line art object, in order that a meta-bit value may control the object optimization laser power selected subsystem 1724 and may form the cutting tool's image data, a high laser power set point is used for a meta-bit value. Or when directing that the cutting tool's image data is in a halftone object, in order that a meta-bit value may control the object optimization laser power

selected subsystem 1724 and may form the cutting tool's image data, a low power set point is used for a meta-bit value.

[0052] It should be understood that the laser power relevant to each of these set points may be set up in the relation of other set points. for example, a low set point -- all -- others -- it is independently set up from a set point. Next, a high set point is set up more highly 10% than a low set point. Or a set point may be independently set up from each other.

[0053] Moreover, many printers are "write-in black" types, and he should understand that they expose the field of the existing media which may be developed marks like a text / line art. If light becomes higher, a mark will become thicker by exposure. Other printers are "white write-in" types which expose the field surrounding a mark, these exposure fields are not developed but an unexposed field is developed. Low exposure is required for creating a thicker mark in a white write-in system by the white field surrounding a text / line art.

[0054] Furthermore, when a print system has other synchronous data sources like an input scanner, since close mixes these data streams with the print data to which it comes from the IOT controller 160 through a data multiplexer 1730 in advance of a print, a meta-bit channel may be used. It may be used in order that the matching equipment may perform mixing in the IOT controller 160.

[0055] In the gestalt of implementation of instantiation of the equipment by above-mentioned this invention, a meta-bit chooses the usable line of a subset which controlled said two or more image-processing subsystems 1712-1738 by the base the whole object, and was specified in each of an object optimization image-processing module and the print multiplexer 1712-1730 by choosing between the one-set meta-bit mapping registers 1740 with which the output is loaded beforehand. For example, when 4 meta-bit is performed, they may be used, choosing them from 16 one-set registers. It may be the thing of all required sizes to be fully able to control these registers respectively and to be chosen from the subset of an image-processing subsystem and the print multiplexer 1712-1730 on the other hand. Thus, the semantics of each metabit value is completely programmable, and may be simply changed by changing the content of the register which it chooses. On the other hand, the register may be chosen from the perfect set of an image-processing subsystem and the print multiplexer 1712-1730 in order to perform a rendering with the highest possibility to the specific type of the object tagged by the meta-bit.

[0056] It should be understood that various image-processing subsystems and print multiplexers 1712-1730 may cover the whole object optimization print system by turns, and may appear on many of other points. For example, after an object type is determined, an object optimization image-processing subsystem and the print multiplexer 1712-1730 can be arranged in IOT170 as some IOT controllers 160 at any time to object optimization ESS 100 like [within the PDL decomposition means 130 and the IOT command instruction generating means 140], so that it may be illustrated. Moreover, these subsystems can be distributed to the whole system.

[0057] Furthermore, these object optimization image-processing subsystems and the print multiplexer 1712-1730 may be performed in hardware, software, or these two combination. In all cases, the subsystem has chosen the data based on the type of an object which a different procedure and different they are processing. Therefore, a different processing result to a different object type is generated.

[0058] Moreover, the meta-bit information connected with a measurement calibration print is used for the color measurement device 190, with the object optimization base, the image-processing subsystem 1712-1722 can be adjusted automatically, and it can stabilize it. For example, as shown in drawing 35, the output color measurement device 190 which measures a calibration print outputs the signal which directs the actual condition of a calibration print. Based on the output signal from meta-bit information and an output color measurement device, the object optimization output correction subsystem 1732 adjusts one tone playback curvilinear (TRC) look-up table from the set of the whole in the object optimization tone playback curvilinear subsystem 1716. Since the object optimization output correction subsystem 1732 is controlled by the tag encoded by the meta-bit channel, the corrections performed to an image processing system function (function) differ to a different object.

[0059] Drawing 3 shows the gestalt of implementation of one instantiation of multi-channel 1630.

As shown in drawing 3 , it connects with the meta-bit unpacker 1631, and the meta-bit FIFO 1626 changes a 8-bit width-of-face meta-bit into at least two 4-bit width-of-face meta-bit nibbles. The output from the meta-bit unpacker 1631 is connected to each of four a color / meta-bit registers 1634-1637. The output from the meta-bit unpacker 1631 depends on the load enabling (LE) signal outputted with FIFO control, and the command / mask channel processor 1680, and is loaded to a color / meta-bit register 1634 thru/or one of the 1637. Similarly, the data input from the fixed color channel FIFO 1624 is offered four a color / meta-bit registers 1634 thru/or 1637, and is loaded to one of a color / the meta-bit registers based on a load enable signal.

[0060] The 18-bit width-of-face input data from the sample channel FIFO 1620 is inputted into sample channel unpacker 1632A, 2-byte width-of-face sample color data and 2-bit "validity-cutting tool" data are changed by the single cutting tool output of two pieces, and by outputting them to internal sample FIFO 1632B continuously, those corresponding "valid bytes" are canceled, when directing that this cutting tool is invalid.

[0061] The direct input of the 2-byte lateral-spreading force from the command instruction FIFO 1628 and the 1-bit lateral-spreading force from a mask FIFO 1622 is carried out to a command / mask channel processor 1680. Moreover, the page synchronization, line synchronization, and IOT clock which are received from IOT through concurrency input port 1650 are inputted into a command / mask channel processor 1680. A color / meta-bit multiplexer 1633, the output multiplexer 1639, and four read-out signals to FIFO 1620-1628 are outputted for a command / mask channel processor 1680 control signal. Four read-out signals consist of a color / meta-bit read-out (RD) signal, sample read-out signals, mask read-out signals, and command instruction read-out signals. Shortly after corresponding FIFO 1620-1628 receives one of the signals of these from a command / mask channel processor 1680, corresponding FIFO (an unit or plurality) will read the next data on a corresponding channel. FIFO (an unit or plurality) 1620 thru/or 1628 fill them up with the data by which reading appearance was carried out from the channel data structure (see drawing 35) which prevented dry cleaning (dry) running by the bus interface 1610, and was stored in memory 150 according to the calculated priority.

[0062] Based on the command instruction received from the command channel FIFO 1628, a command / mask channel processor 1680 generates load enable signal LE to one to which a color / meta-bit register 1634 thru/or 1637 correspond. If it enables one of a command / the mask channel processors 1680, it loads the 9-bit width-of-face fixed color data from the fixed color FIFO 1624, and the meta-bit data to 4 bits from the meta-bit unpacker 1631 simultaneously. Or color data may be independently loaded from a meta-bit by the separate command. However, since they are loaded simultaneously continuously, as for a color / meta-bit register 1634 thru/or 1637, it is desirable that it is a register according to individual always [both] referred to.

[0063] Moreover, a command / mask channel processor 1680 generates a control signal to a color / meta-bit multiplexer 1633, in order to choose the color / meta-bit register 1634 which should be inputted into the output multiplexer 1639 and the meta-bit register 1638 thru/or one of the 1637. Furthermore, a command / mask channel processor 1680 generates a control signal to the output multiplexer 1639, in order to choose between the output from a color / meta-bit multiplexer 1633, and the output from internal sample color FIFO 1632B. The output multiplexer 1639 outputs the 8-bit width-of-face color data outputted by a color / meta-bit multiplexer 1633 and the metadata of the 1st bit. This is inputted into data FIFO 1642. Simultaneously, the meta-bit register 1638 stores the 2nd - the 4th meta-bit, and outputs them to the output meta-bit FIFO 1640. It should be understood that it is unnecessary in the thing [as / in the gestalt of desirable operation] containing all the 4-bit width-of-face meta-bit nibbles. Or the number of the meta-bits offered to IOT may be 1, 2, 4, or 8. Thus, the gestalt of implementation of instantiation shown in drawing 2 and 3 should not be interpreted as what restricts the number of the meta-bits offered to IOT to 4.

[0064] Drawing 4 shows internal functional block of a command / mask channel processor 1680. In drawing 4 , only the control line is shown except for the command channel line and the mask data channel line. In the command / mask channel processor 1680, two configuration registers

1681 and 1684, a register CSR 0, and CSR1, are prepared, respectively. The 1st configuration register 1681 offers control of the feature (function) shown in a table 1.

[A table 1]

ビット	ハイ	ロー
8	将来の使用のためにリザーブ	将来の使用のためにリザーブ
7	白=FF	白=00
6	ノーマルモード	診断モード
5	レジスタ読出し	FIFOs読出し
4	FIFOモードに対してレジスタをオン	FIFOモードに対してレジスタをオフ
3	メタビットFIFOを使用	カラーFIFOの9番目のビットを使用
2	メタビットアンパック1をオン	メタビットアンパック1をオフ
1	メタビットアンパック0をオン	メタビットアンパック1をオフ

構成/ステータスレジスタ 0 (CSR 0) ビット割り当て

[0065] The 1st bit 1 and bit 2 of the configuration register 1681 define 2 (need is accepted or it is 3 or 4) meta-BITTOAN packing method used by the meta-bit unpacker, in order to unpack a meta-bit. That is, when the method of the addition for determining how the cutting tool of meta-bit data is decomposed to meta-bit PAKKETTO is required, it may be used in order that the 1st and 2nd bits may offer [both] the type to four of meta-BITTOAN packings.

[0066] When the 1-bit width-of-face meta-bit channel of one ** is offered to IOT, the bit 3 of the 1st configuration register 1681 is used in order to make the meta-bit FIFO disenabled (an activity is impossible). The 1st bit 4 and bit 5 of the configuration register 1681 are used by the diagnostic mode, and read data from FIFO and an internal register. The bit 6 of the 1st configuration register 1681 is used in order that a multi-channel combiner may direct normal mode or the diagnostic mode. The bit 7 of the 1st configuration register 1681 is used in order that that all white (for example, natural background color of paper) is offered may direct the data byte which consists of 0, or the data byte which consists of 1 altogether. The 8th bit of the 1st configuration register 1681 is not performed.

[0067] When the bit 6 of the 1st configuration register 1681 is a low, the multi-channel combiner 1630 is the diagnostic mode. In the diagnostic mode, all of the internal register of the multi-channel combiner 1630 may be examined. For example, the content of color / meta-bit register 1634 thru/or 1637, and sample FIFO1632B is examinable on real time. By setting the diagnostic SEL line shown in a table 2 to the value of 0-4, a SEL line can send the value of the selected register to data FIFO 1642 compulsorily. As shown in a table 2, the value 5-14 of a SEL line is used for read-out of other registers of the multi-channel combiner 1630, and writing for the further diagnostic information.

[A table 2]

SEL	アクセスされるレジスタ
0	ビット0FIFOへのカラー/メタビットレジスタ0
1	ビット0FIFOへのカラー/メタビットレジスタ1
2	ビット0FIFOへのカラー/メタビットレジスタ2
3	ビット0FIFOへのカラー/メタビットレジスタ3
4	ビット0FIFOへのサンプルレジスタ
5	リード/ハックハスに配されるSR0
6	リード/ハックハスに配されるSR1
7	リード/ハックハスに配される又は書込まれるCSR0
8	リード/ハックハスに配される又は書込まれるCSR1
9	リード/ハックハスに配されるメタビット出力レジスタ
10	リード/ハックハスに配される又は書込まれるビットステータスコントロー (0:7)
11	リード/ハックハスに配される又は書込まれるビットステータスコントローハイ (8:12)
12	リード/ハックハスに配される又は書込まれる先端白カウンタ (0:7)
13	リード/ハックハスに配される又は書込まれる後端白カウンタ (0:7)
14	リード/ハックハスに配されるフカフカレジスタ
15	将来使用のためにリザーブ

診断SEL値に対するレジスタアクセスデコード表

[0068] When the bit 6 of the configuration register 1681 is yes (namely, normal mode), the multi-channel combiner 1630 generates the Normal print data, and it is transmitted to it to data FIFO 1642.

[0069] A table 3 shows bit assignment of the 2nd configuration register 1684. A bit 7 and a bit 8

provide an IOT interface with flexibility by determining the polarity of an IOT interface signal. A bit 5 and a bit 6 offer diagnostic exchange. Bits 1, 2, 3, and 4 are not performed with the gestalt of implementation of this instantiation.

[A table 3]

ビット	ハイ	ロー
8	P同期立下り	P同期立ち上がり
7	L同期立下り	L同期立ち上がり
6	偽P同期オン	偽P同期オフ
5	偽L同期オン	偽L同期オフ
4	将来使用のためリザーブ	将来使用のためリザーブ
3	将来使用のためリザーブ	将来使用のためリザーブ
2	将来使用のためリザーブ	将来使用のためリザーブ
1	将来使用のためリザーブ	将来使用のためリザーブ

構成ノステータスレジスタ1 (CSR1) ビット割り当て

[0070] If drawing 4 is referred to again, a command / mask controller 1686 will receive a 16-bit command instruction and a 1-bit mask channel input. Normal / repeat mode bit sets tables 4 and 5 to Normal -- having (that is, a bit 15 being set to 0) -- the bit assignment to a 16-bit command instruction is shown.

[A table 4]

コマンドビット	フィールド記述子
0	RepCnt0
1	RepCnt1
2	RepCnt2
3	RepCnt3
4	RepCnt4
5	RepCnt5
6	SCRes0
7	SCRes1
8	SrcB0
9	SrcB1
10	SrcA0
11	SrcA1
12	Cntrl0
13	Cntrl1
14	LdColor
15	Repeat/Normal

ノーマルモード (ビット15=0)

[A table 5]

LdColor		コマンド
0		カラーをロードしない
1		SrcB カラー/メタビットレジスタをFIFOsからロード 注:1) "メタビットモード"を使用の場合、 メタビットがアンパッカーからロードされる。 注:2) カラー/メタビットレジスタは、これがカラー/メタ ビットレジスタ0乃至2に対するこのコマンドの第1の クロックサイクル又はカラー/メタビットレジスタ3のこのコマンド のあらゆるクロックサイクルである時、ロードされる。
Ctrl1	Ctrl0	コマンド
0	0	常にSrcB
0	1	常にSrcA
1	0	マスクがSrcAとSrcBとの間で選択
1	1	マスクがSrcBとSrcAとの間で選択
SrcA1	SrcA0	コマンド
0	0	SrcAがカラー/メタビットレジスタ0を選択
0	1	SrcAがカラー/メタビットレジスタ1を選択
1	0	SrcAがカラー/メタビットレジスタ2を選択
1	1	SrcAがサンプルを選択
SrcB1	SrcB0	コマンド
0	0	SrcBがカラー/メタビットレジスタ0を選択
0	1	SrcBがカラー/メタビットレジスタ1を選択
1	0	SrcBがカラー/メタビットレジスタ2を選択
1	1	SrcBがカラー/メタビットレジスタ3を選択
SCRes1	SCRes0	コマンド
0	0	サンプリングクロックを4で割る
0	1	サンプリングクロックを3で割る
1	0	サンプリングクロックを2で割る
1	1	サンプリングクロックを1で割る
RepCnt0-5		コマンド
		ノーマルコマンド・リピーターカウンタ

ノーマルコマンドデコード表

[0071] In normal mode, as shown in a table 4, a bit 0-5 expresses 6 bit-count value. As for counted value, that it is the pixel which this command follows directs how many it is effective. Depending on a value [as opposed to bits 6 and 7 in a table 5], a sample clock is divided by 1, 2, 3, or 4. By breaking a sample clock, when the resolution of an output image carries out the repeat output of the current color of internal sample FIFO1632B, it may change. Namely, when both the values of the bits 6 and 7 as which a crack" command is chosen by 4 in "sample clock are 0, the 4 times repeat output of the internal sample FIFO1632B is carried out for every clock cycle. It follows, for example, the printer of 400 spots / inch (spi) resolution does not have the need, and processing of in any way an addition can print a sample map by 400spi(s), 200spi, 133spi, and 100spi(s).

[0072] A color register-select B value is expressed, and, as for bits 8 and 9, bits 10 and 11 express a color register-select A value. As shown in a table 5, as for a color register-select A value and a color register-select B value, any of a color / meta-bit register 1634-1637 (or internal sample FIFO1632B) direct whether it is passed by the combiner to data FIFO 1642. . which a color register-select A value chooses between a color / megabit registers 1634, 1635, and 1636, or the internal sample FIFO, and a color register-select B value chooses between a color / meta-bit register 1634-1637 — there are two selection values A and B, and, thereby, a 1-bit channel is used for the change between them. This mode is specified by bits 12 and 13, and it is used in order to switch promptly between two registers specified without the instruction overhead of any addition by A and B.

[0073] Bits 12 and 13 express a 2-bit combiner control value. A combiner control value directs

the color data which should be outputted to data FIFO 1642 by the multi-channel combiner 1630 to the following n pixels directed with 6 bit-count value. As shown in a table 5, it is directed whether a combiner control value comes from the combination of the register with which the color which should be outputted is chosen by a color register-select A value or the color register-select B value, or its two registers which are further controlled by the mask channel.

[0074] Finally, as shown in a table 5, a bit 14 directs whether the following color data outputted from the color channel FIFO 1624 should be loaded to one of a color / the meta-bit registers 1634-1637 by the color register-select B value.

[0075] In actuation, if a new normal mode command is inputted from a command channel FIFO 1628, the load color bits 14 will be decoded, in order that the following 8-bit color data byte stored in the color channel FIFO 1624 may determine whether it should be loaded to the color / meta-bit register 1634-1637 directed by color register-select B bit 8 and 9. When color register-select B bit 8 and 9 directs a color / meta-bit register 1634-1636 (a color / meta-bit register 0-2), a color is loaded from the color channel FIFO 1624 only by the 1st clock cycle of a command.

[0076] However, color register-select B bit's 8 and 9 directions of a color / meta-bit register 1637 (a color / meta-bit register 3) load new fixed color data to a color / meta-bit register 1637 from the color channel FIFO 1624 for every clock cycle of this command. Thus, a fixed color channel may be actually used as a low-speed sample channel. In the high image of some complexity, since the fixed color data in a scan line change promptly (that is, 1 - 8-pixel all are changed), they are that, and they generate a new command word for every new color, and it stores, and transmits, and processing required in order to decode becomes superfluous, consequently a cutting tool map expression is faced and it becomes negative compression.

[0077] This effective negative compression may be avoided, when the string of a fixed color is offered to the fixed color channel FIFO 1624, one fixed color cutting tool of the string of a color FIFO 1624 to a fixed color is loaded to read-out and it loads it to a color / meta-bit register 1637 for every clock cycle. Since the fixed color channel FIFO 1624 can load 2 bytes at once as in the sampled data channel FIFO 1620 not at 4 bytes but at 1 time, as for using the fixed color channel FIFO 1624 in this way, the throughput of a fixed color channel becomes lower than the throughput of a sample data channel. However, the this "slow sample" method avoids a required overhead, when generating the DMA pointer used for a regular sample data channel (explained in full detail below). Thus, to the short stretch of the fixed color which changes promptly, a this "slow sample" channel is not used to a big image, although it is the most useful.

[0078] Next, the combiner control bits 12 and 13 are decoded in order to determine whether it is used in order that the mask data from a mask channel may control the data flow to data FIFO 1642 further from the multi-channel combiner 1630, while determining any of the color selection register banks A and B are used, in order to determine any shall be used as a color data source between a color / meta-bit register 1634-1637 (or internal sample FIFO1632B). Of course, if it directs that the load color bits 14 and color selection register B bit 8 and 9 use a color / meta-bit register 1637 as a "slow sample" color channel, control bits 12 and 13 must direct to use a color register-select B bank. On the other hand, in order that the command bits 12 and 13 may direct a color data source, when it directs to use the color register-select bank A, the load color bits 14 and color register-select bank B bit 8 and 9 may be used in order PURIRODO [the following color data word which is not directed by 10 and 11 A bits of color register selects, and is outputted to one of a color / the meta-bit registers 1634-1637 from the fixed color channel FIFO 1624]. Thus, PURIRODO [the following fixed color data word / one of a color / the meta-bit registers].

[0079] Next, it is decoded in order to determine any shall be chosen depending on any of the register-select banks A and B are directed by the combiner control bits 13 and 14 between a color / meta-bit register 1634-1637 (or internal sample FIFO1632B) as a color data source with which 10, 11, or color register-select B bit 8 and 8 is outputted A bits of color register selects. Next, if the combiner control bits 12 and 13 direct the color register-select bank A and 10 and 11 direct to use the data stored in internal sample FIFO1632B A bits of color register selects, the sampled color resolution bits 6 and 7 will be decoded in order to determine the factor which

divides a sample clock. Next, directions of that the combiner control bits 12 and 13 control an output color further using a 1-bit mask channel switch the output to data FIFO 1642 between the color register (or the sample FIFO) chosen by 10 and 11 A bits of selections, and the color register chosen by selection B bit 8 and 9. Finally, a command instruction current in the repeat count bit 0-5 is decoded in order to determine how many a pixel is effective.

[0080] Therefore, a color / meta-bit register load selection 1685 is controlled by the value of the load color bits 14 and color register-select B bit 8 and 9 to be shown in drawing 4 . Similarly, a color / meta-bit multiplexer selection controller 1689, and the output multiplexer selection controller 1692 are controlled by combination of 10, 11, and a mask channel as mentioned above the command control bits 12 and 13, color register-select B bit 8 and 9, and A bits of color register selects.

[0081] However, if the repeat mode bit 15 is set to a high, the gestalt the command bit of a command instruction is instructed to be to tables 6 and 7 will be taken.

[A table 6]

コマンドビット	フィールド記述子
0	RepCnt0
1	RepCnt1
2	RepCnt2
3	RepCnt3
4	RepCnt4
5	RepCnt5
6	RepCnt6
7	RepCnt7
8	RepCnt8
9	RepCnt9
10	RepCnt10
11	RepCnt11
12	RepCnt12/Mask Scanline Disable
13	RepType0
14	RepType1
15	Repeat/Normal

繰り返しモード (ビット15=1)

[A table 7]

RepType1	RepType0	コマンド
0	0	最後のノーマルコマンドを繰り返す
0	1	白出力: 先端白データのために使用。繰り返しカウンタが0に等しくあるべきという点に注意。
1	0	ライン端(End of Line): 後端白データのために使用。繰り返しカウンタが0に等しくあるべきという点に注意。
1	1	ページ端(End of Page)
	Rep12	コマンド
	0	注: ライン端がこのコマンドからリポートされると、Rep12は、Mask Channel Scanline Disable; もしこのビットが0の時、Mask Channelは次のスキャンラインに対して使用不能である。
	1	ライン端の時、Mask Channelは、次のスキャンラインに対して使用不能である。
	RepCnt0-12	コマンド
		コマンド繰り返しカウンタ(Command Repeat Count)を繰り返す。

繰り返しコマンドデコード表

[0082] As shown in tables 6 and 7, the command bit 0-12 expresses a repeat count. Since the repeat count in repeat mode offers the range of a repeat count 128 times in normal mode,

repeat mode is used when it has the same color data or the same object like the image with which the very broad structure of a scan line was scanned, or a big fixed color field.

[0083] Finally, bits 13 and 14 express the repeat type of a repeat mode command instruction. When a repeat type is normal, a front command is repeated to the number of pixels of the addition directed by the repeat count bit 0-11. In this case, the command register 1682 before being shown in drawing 4 is used. A repeat command type's directions of the edge of head white data or after [a line] white data load one single white cutting tool to data FIFO 1642 by the white (transparency) generator 1690. This white (transparency) generator 1690 is set to white correction value with the configuration register 1681. If a repeat command type directs the edge of the page, this command will suspend activation of the further command until a circuit is reset.

[0084] Only when the repeat type bits 13 and 14 direct a line type edge, a bit 12 should be redefined so that it may be mask scan line enabling / disenabled bit, and being used since the bit turns on or turns off a mask channel to the following scan line should be understood. Thereby, only the scan line which has these related mask data is allotted to a mask channel, and compresses the data which needs it for a mask channel. This is shown in a table 7.

[0085] The repeat type bits' 13 and 14 directions of an output white type or line type edge should also understand that a repeat count is not used. Instead, the white space of the start of a line and an end is the special case of video-data generating with the advantage which increases the amount of time amount which can be used since actual page data is generated. PURIRODO [three counters are attached in the IOT interface controller 1691, and / data with these exact at the beginning of each color separation] since a head and back end white (transparency) data are generated. The head white counter 1694 is used in order to count the initiation margin of each scan line. A count will be started if a line synchronizing signal with new line and page synchronous (sync) handler 1688 is received from IOT. While the head white counter 1694 has counted, the IOT interface controller 1691 carries out the repeat output of the single white cutting tool who made it activity impossible to read-out enable [of data FIFO 1642], and was allotted to data FIFO 1642 by the head white repeat command. Repeat reading appearance of this white cutting tool is carried out by IOT170. If the head white counter 1694 amounts to 0, the video-data counter 1695 starts a count, and the IOT interface controller 1691 will make data FIFO 1642 and the output meta-bit FIFO 1640 enable (usable), and will pour those data to IOT170 under the synchronization from the IOT controller 1691. If the video-data counter 1695 amounts to 0, the back end white counter 1696 will start a count, and the IOT interface controller 1691 will carry out repeat reading appearance of the single white cutting tool allotted to data FIFO 1642 by IOT170.

[0086] The IOT interface controller 1691 operates independently from the portion made to fill up with the data of FIFO control, and the command / mask channel processor 1680, and the output meta-bit 1642 and FIFO 1640. The IOT interface controller 1691 receives a clock signal, a page synchronizing signal, and a line synchronizing signal from IOT170 through a line and the page synchronous handler 1688, and returns a return clock signal and a read-out data signal to IOT170. As long as it has the effective data with which the multi-channel combiner 1630 was filled up with data FIFO 1642 and the output meta-bit FIFO 1640, the IOT interface controller 1691 is used in order to synchronize read-out of the data by exact time amount by IOT.

[0087] As shown in drawing 5, a bus interface 1610 has burst FIFO 1611, the sample channel control 1612, the slow channel control 1613, a register 1614, a decoder 1615, buffer memory 1616, the channel mediation logic 1617, and the card controller 1618.

[0088] Basic actuation of a bus interface 1610 is searching PAKKETTO on the bus which uses a direct memory access controller (DMA), and writing them in one of the five FIFO channels 1620-1628. A bus interface 1610 is once programmed in front of each page, and since it is filled up with the FIFO channel 1620-1628 next, it operates independently. The channel mediation logic 1617 is used in order to determine the priority with which each of the FIFO channel 1620-1628 is filled up. This is based on the signal from each of FIFO 1620-1628 generated when it is almost empty, programmable "weight" to each channel, and the interleave option to three of these channels. Weight directs the count of the line which remains as priority with the present highest

channel. The mediation method of the gestalt of operation of the 1st of a bus interface 1610 is based on three subsets of priority. The priority subset 1 is the always highest priority, and it includes drawing of a new sample and a mask channel pointer so that it may be explained in full detail later. Priority carries out alternation of the priority subsets 2 and 3. The priority subset 3 of the priority subset 2 is a round robin drawing command including sample channel data, and it takes out a mask, a color, a meta-bit, and command channel data.

[0089] The reason for a background of this mediation method is as follows. Although a pointer (subset 1) is small and there are not, as stated below, it is important drawing. [many] Sample channel data (subset 2) is the raw cutting tool who may demand the biggest bandwidth through a system. Other channels are called a "slow channel." The reason is that it has width of face with few those channels FIFO 1622-1628 than 32-bit width of face. If they are taken out, it is required to write the taken-out WORD in burst FIFO 1611, and, thereby, the slow channel control 1613 can unpack them on the FIFO channel 1622-1628 according to those individuals in parallel. Between this actuation, other burst ejection may occur in parallel to the sample channel FIFO 1620.

[0090] The burst transmission size of drawing is controllable to each channel. One of the registers 1614 stores the burst size to each channel. Moreover, there is an one-set register of the register 1614 which stores the address in the memory 150 which the data to each channel starts. Since memory size is saved, the offline processing (indirection) of 1 level is used for a mask channel and a sample channel in data ejection. Instead of using a single address-pointer register, each of a mask and a sample color channel has three registers. The 1st thing points at the address in the memory 150 in which the table (table) is stored. These each of a pair of specifies the portion of the memory 150 by which a next sample map or a next bit map section is found including the list of the address / size pairs. [table / this] The sample channel control 1612 takes out the next address to the sample address register of a register 1614, and takes out the size of the block of drawing and sample data to the sample size register of a register 1614. Next, it can be taken out from the portion of memory 150, and it takes out the following address / size pair in a sample data table after that until the data of an amount with an exact bus interface 1610 is taken out. A mask channel is processed similarly.

[0091] Since it generates continuously in the word boundary, in the case of the sample data which the data of the 1st actual valid byte generates in WORD, drawing is possible. Thus, the sample channel control 1612 compares the word address in the memory 150 actually taken out in the byte address in the memory 150 which was going to be taken out, and must be able to tag it to 3 bytes of the 1st WORD taken out with the "invalid" tag. Sample channel unpacker 1632A does not load these invalid cutting tools to internal sample FIFO 1632B, and can cancel them later.

[0092] Other pairs of a bus interface 1610 contain the decoder 1615 which processes the buffer memory 1616 for making a data rate equal, the card controller 1618 which processes a low-level S bus (Sbus) card function, S BASURI QUEST address mapping, and decoding. One important function of a decoder 1615 is offering the interface to the serial port of IOT170 which transmits status information and can receive.

[0093] Drawing 6 shows the gestalt of implementation of instantiation of the 2nd of the IOT controller 260. As shown in drawing 6, the bus interface 2610 with same bus interface 1610, essential actuation, and configuration receives a 4-byte burst from a bus 114, and outputs a 32-bit data stream to a compressor (compressor) / decompression (compression discharge machine) controller 2650, and the integrated combiner / channel FIFO 2630. An integrated combiner / channel FIFO 2630 unifies the multi-channel combiner 1630 with a data channel FIFO 1620-1628. A compressor / decompression controller 2650 is controlled by meta-bit information transmitted from memory 150 while it enables the activity of the data compressed when transmitting data to the IOT controller 260 from memory 150. A compressor / decompression controller 2650 enables meta-bit information to direct which type of compression was applied to the block with which data was received in this way.

[0094] Therefore, a data compression can be optimized based on an object type (that is, is data other data types which may be compressed the optimal using the color picture data compressed

the optimal using the "JPEG" (joint photographic expert group) technique, white / black bit map data and run length encoding, or a binary compression technique like CCITT?). Therefore, based on the object type of the data inputted into a compressor / decompressure controller 2650, data is outputted to a sample data compressor / decompressure 2620, or the binary sample data compressor / decompressure 2640. Furthermore, the scanner interface 116 (you may be an interface to other devices like a digital camera) can be attached in a compressor / decompressure controller 2650, and it is real time, and thereby, the on-the-fly data from a scanner is obtained and compressed, it is transmitted to a compressor / decompressure controller 2650, compression discharge is carried out, and it may be inputted into the suitable channel of an integrated combiner / channel FIFO 2630. On-the-fly insertion of the portion of an image is enabled by this through the scanner or the object of other equivalence like the scanner of a copying machine, or an integrated copying machine / printer, and thereby, on-the-fly data does not need to change PDL description of a page image, and may be combined under control of a meta-bit channel to a page image. Furthermore, thereby, on-the-fly optimization of the portion of an image is enabled through a scanner or other equivalent devices. For example, if the rendering of the portion of an image should be carried out using the laser beam of higher level, an operator uses the increased exposure level set point, and can identify the rendering of which portion of an image should be carried out.

[0095] Once compression discharge of a sample or the binary data is carried out, it will be inputted into an integrated combiner / channel FIFO 2630, and it will operate, as substantially mentioned above about drawing 2 -4.

[0096] Furthermore, since the IOT controller 260 contains the color space converter (transformer) 2670, sample color data and fixed color data can be arranged on the optimal color space, before it is stored or transmitted. As shown in drawing 6, an integrated combiner / channel FIFO 2630 outputs 24-bit WORD (per color three colors X 8 bits) to the color space converter 2670, and this converter changes 24-bit 3 color data into 4 bytes of color data, next that data is returned to an integrated combiner / channel FIFO 2630 as 4 bytes showing C, Y, M, and K color detached core. Moreover, color data may be directly inputted into the color space converter 2670 from a bus interface 2610. It may be used in order to choose the optimal color space conversion to the specific object type to which meta-bit data control the color space converter 2670, and current data relates in both cases. As a portion of object optimization color space conversion, an optimal tone playback curve (TRC) may be chosen using a meta-bit. Next, the integrated combiner / channel FIFO 2630 which operates as mentioned above in relation to drawing 24 output output color data and a meta-bit to IOT170. According to the capacity and speed of IOT170, the last data may be outputted in a 8-bit cycle plus 1per output-4 meta-bit, or a 32-bit cycle plus 1per output-4 meta-bit.

[0097] In the gestalt of implementation of instantiation of the 3rd of the IOT controller shown in drawing 7, a compressor / decompressure 3650 is used only in order to control a sample data compressor / decompressure 3620. The binary sample data compressor / decompressure 2640 of drawing 6 were replaced with more advanced structure. In the gestalt of implementation of this instantiation, a binary compressor / decompressure 3640 is in any of an adaptation compressor / decompressure, and a high-speed CCITT compressor / decompressure. However, this IOT controller 360 operates, as generally mentioned above about the gestalt of the 1st of an IOT controller, and the 2nd suitable operation.

[0098] Finally, drawing 8 shows the gestalt of implementation of instantiation of the 4th of the IOT controller 460. In the gestalt of implementation of this 4th instantiation, an integrated combiner and a channel FIFO 4630 output three colors (24-bit WORD) or four colors (32-bit WORD) to the 1st color space converter 4670 and the 2nd color space converter 4675. Since two of 4 color detached cores outputted to IOT170 are generated, each of the 1st color space converter 4670 and the 2nd color space converter 4675 is used. Generally, each of a color space converter operates at speed twice the speed [the combiner integrated, a channel FIFO 4630, and] of packers 4672 and 4677.

[0099] For example, it is the 1st clock cycle, and the 1st color space converter 4670 outputs a 8-bit C color detached core data byte, it is the 2nd clock cycle and outputs a 8 bit M color

detached core cutting tool. Similarly, by the 1st clock cycle, the 2nd color space converter 4675 outputs a 8 bit Y color detached core cutting tool, is the 2nd clock cycle and outputs a 8 bit K color detached core cutting tool.

[0100] By the 1st clock cycle, the 1st color space converter 4670 outputs 8-bit C data to a packer 4672, and the 2nd color space converter 4675 outputs 8 bit Y data to a packer 4677. Similarly, by the 2nd clock cycle, the 1st color space converter 4670 outputs 8 bit M data to a packer 4672, and the 2nd color space converter 4675 outputs 8 bit K data to a packer 4677. Since packers 4672 and 4677 operate in the one half of the speed of the color space converter 4670 and 4675 ** next, they output 16 bit data unified with IOT170 from the 1st and 2nd color space converters 4670 and 4675 in parallel, and offer four color detached cores simultaneously as four 8-bit WORD altogether. Thus, the larger range of IOT170 can use it by this system, and this system contains the one pass, and 4 color copying machines / printer which uses four different print drums, in order to be the single pass of a copy sheet and to form all 4 color detached cores, and C, M, Y and K through IOT.

[0101] The following drawings show the data structure stored in the PDL decomposition means 130, IOT data, the command instruction generating means 140, and memory 150. Memory 150 contains the RAM portion 151 and the non-volatile portion 152. The non-volatile portion 152 may consist of either of the memory devices of other equivalence which offer a hard disk, a dismountable tape, a floppy disk, an optical disk, a flash memory, and the non-volatile data storage over a long period of time. As shown in drawing 9, after starting at step S10, a document is created at step S20 by document creator like an operator with a graphical artist or the technology of other desktop publication programs.

[0102] Once a creator ends document creation at step S20, the print data for IOT will be prepared at step S30 using object assignment or object optimization compression, and a rendering technique. Preparations of the print data in step S30 are made by the PDL decomposition means 130, IOT data, and the command instruction generating means 140. Preparation of print data stores it in the RAM portion 151 of memory 150 using the data structure shown in drawing 28 -31.

[0103] Next, after print data is prepared at step S30, in step S40, it is determined whether the data structure of the RAM portion 151 of memory 150 should be stored in the non-volatile portion 152 of memory 150. **** with the storing step S50 a document creator may specify the storing step S50, and big [a print system] -- you may require that the resource of a complicated document should be saved. Decision of saying [that the data structure stored in the RAM portion 151 of memory 150 should be memorized] stores a nonvolatile memory portion in the compressed page at step S50. After determining that the compressed page to this document stored by S50 should be printed, this requires inevitably that the page compressed as in step S60 should be searched, when a creator or a print system copies them to the RAM portion 151 from the non-volatile portion 152 of memory 150. Thus, the print data prepared at step S30 is outputted to step S70 irrespective of whether it was stored at step S50 and was again called at step S60, or it was directly transmitted by step S40.

[0104] Moreover, IOT170 is not driving actively step S20-S50, step S110-130, and those substeps described later, therefore since real-time constraint does not restrict these steps, it should be understood that it is not necessary to perform on real time. Reversely, since IOT170 is driving step S70-S100 actively, it should be understood that it should perform on real time. Therefore, processing of the data in real time and the page to IOT170 on which failure of offer is printed by IOT are made into incorrectness. Depending on the type and capacity of the nonvolatile memory portion 152, step S60 does not have to be carried out, even if it performs on real time.

[0105] In step S70, the print data prepared at step S30 is combined and printed using object assignment or object optimization compression discharge, and a rendering technique. Simultaneously, at step S80, real-time data is captured and synchronizes with the print data prepared at step S30. The real-time data of step S80 is captured from other sources like other devices which generate a scanner, the scanner portion of a copying machine, a digital camera, a remote computer, or data, and can be transmitted to the IOT controller 160 on real time.

[0106] Next, at step S90, a creator or a print system determines whether this print actuation is a calibration print. A creator or automatic calibration processing of IOT170 is step S90, and if it determines that this print is a calibration print, a print page will be automatically measured on real time in step S100, in order to determine whether object optimization rendering accommodation needs to correct which rendering of the object of a document. For example, the IOT controller 160 outputs a predetermined test document to IOT170 during the calibration test of one specific type. This test document includes the test patch by which a rendering is carried out, in order to simulate an object of a specific type like a pictures target or a photograph-- (sampled) object. the inside of IOT170 after IOT170 prints a test document -- or a mounting **** sensor measures the various colors of a test document to it, and provides it with the measurement data to the IOT controller 160. Being printed on a test document to the type of the actual color and object which were printed as directions by measurement data compares the color meant, and it adjusts the IOT controller 160. In this way, the tone playback curve drift of IOT170 produced by change in temperature, humidity, or other environmental factors may be corrected with the object optimization base by [like a fixed color object, a sample image data object, or a color text object] changing a tone playback curve to a different object type. Thus, a calibration print is measured, and object optimization rendering accommodation is made at step S110, and control returns to step S70 for printing a actual document.

[0107] However, if this print is not a calibration print at step S90, although control progresses to step S110 and it is not real time, as for a creator, that printed document will determine whether to be O.K. or not there. When the printed document is not O.K., control progresses to step S120, edit of the object in the print data based on a creator is attained, and control returns to step S30. In step S30, an object may be adjusted through the operator interface of the object optimization ESS 100 in advance of smoothing (it makes an object type lose generally). Since it is not necessary to return to the original non-decomposing PDL document file and an operator or a creator can perform color space conversion (an unit or plurality), tone playback curve (an unit or plurality), and/or slight accommodation in other factors, the great portion of processing required in order to prepare print data using object optimization compression and a rendering does not need to be repeated. furthermore -- since an object type is held at this point -- these types of correction -- each -- ** -- it is not repeat-like and is once made by the scan line to an object.

[0108] This edit step S120 that uses the information which can be used by the object optimization ESS 100 When processing change of the rendering relation to a document, a creator The step which calls again the document of the whole needed even when it is required only for the step which returns to a workstation, and few portions to change to a workstation display, The step which changes a document, the step which generates the new PDL version of a document, Before seeing the step which resends a document to a printer, and a new print, it differs from the Prior art in that the step which the time amount of the step which waits for a document to pass decomposition processing requires is avoidable. Before only the substep of the last of the preparation print data-processing step S30 can see a print with a new creator by the ability of rendering modification to each object on a page like modification to mid tone cyanogen separation of the specific pictorial object on a page inputting into a printer instead of, it is necessary to perform again within the object optimization ESS 100. Since the last stage of document generation is often advanced repeat processing, the amount of save of time amount is quite large.

[0109] Furthermore, the rendering control which can use a document creator by workstation Since the color on which the direct control which can often be offered by the object optimization ESS 100 differs, and it gets down, and a color printer prints a workstation display indicates that a color differs A document creator is not by editing by workstation of a Prior art, and converging quickly with the desirable color to the object in a document is expected by using the high-speed reprint step S120 accompanied by edit.

[0110] however, a certain modification whose print is O.K. and which control progressed to step S130 from step S110, and was made at step S120 there when becoming -- the object optimization ESS 110 -- or it is saved to a standard format by document generated software for

automatic inclusion on a document. Next, control progresses to step S140 and processing ends it there.

[0111] Drawing 10 shows the processing for preparing print data using object optimization compression and the rendering of step S30 of drawing 9 more to details. As shown in drawing 10, print data-preparation processing of step S30 is started at step S200. In step S200, the next page (or the 1st) of the document generated at step S20 is obtained from the internal PDL file source means 110 or other remote PDL file source means 12 as a current page.

[0112] Next, in step S210, the object list of a current page is constituted and the object optimization rendering tag related in the specific object to this page is incorporated. In step S210, an object optimization rendering tag can be automatically generated from the object type determined to each object by the PDL decomposition means 130 and/or IOT data, and the command instruction generating means 140. Or while the automatic processing for generating an object optimization rendering tag is used as a default mode, when preparing a document at step S20, a document creator can contain clearly the object optimization rendering hint when specifying an object using PDL. These creator insertion rendering hints Defining an object type clearly, color space conversion, or a tone playback curve is defined clearly, Setting halftone screen frequency and/or an angle clearly and when that is not right The maximum laser power modulation which directs the priority over GAMUTO mapping usually automatically set up by the PDL decomposition means 130 and/or IOT data, and the command instruction generating means 140, and/or other desirable object optimization rendering parameters, namely, the thing for which a set point is set clearly -- it contains.

[0113] Since the object list containing an object optimization rendering tag is generated to the current page in step S210, in order to guarantee that they were not canceled, the monitor of a system resource like the memory which can be used in step S220 is carried out. For example, a very complicated page like the complicated clipping region which has the sweep set up with a certain angle to the direction where a page intersects perpendicularly needs many objects for a degree with the inadequate memory resource of the RAM portion 151 of memory 150 in an object list. That is, this page includes the problem of navigation compression actually.

[0114] Thus, the fall-back mode which does not use another memory for every object of each addition is offered. In this fall-back mode, the rendering of the current page may be carried out in the resolution which the rendering was carried out to the sample channel, and fell by the system resource. If step S220 determines that a resource is not enough, control will progress to step S230 and it will prepare the print data to a current page using fall-back mode. Generally, it depends for fall-back mode on generating the conventional bit map / cutting tool map used for conventional IOT170. in this case, as for a page, the print parameter for every object is optimized -- as -- although not printed, that page may be printed at least. Furthermore, since some PDL defines the print parameter to the present page by processing overflow of a resource with reference to a front page (that is, a page not being independently) by carrying out a default to the conventional bit map / cutting tool map, it is possible to perform such reference to the parameter [as / in step S230] with which the defect page was defined above.

[0115] In step S230, attainment of the page which comes out of an available resource may start the conventional PDL decomposition processing from the 1st page of a current document. All the conditions and graphical operator (operator) parameters are reset by 0. The graphical operator from the 1st page is processed only in the limitation which maintains a graphical condition to accuracy. That is, a bit map or a cutting tool map is not written in. This continues to the independent page of the last before meeting with a defect page. Although both a graphical operator and image data are processed from the point, image data is not outputted to IOT. If a defect page is reached, the processed image data will be outputted to IOT. Processing of the continuing page and the output to those IOT(s) continue to the independent page of the beginning after meeting with a defect page.

[0116] From this point, the restart of the object optimization PDL decomposition means 130, IOT data, and the command instruction generating means 140 is carried out by the 1st page of a current document, and they reset all data fields and graphical operators to 0. The object optimization ESS 100 continues object optimization processing, without generating image data to

the independent page of the beginning after the defect page was discovered. From this point to the following defect page, the PDL decomposition means 130, IOT data, and the command instruction generating means 140 operate as mentioned above. Next, this processing is repeated for every defect page until the whole document is printed.

[0117] Or in step S230, two parallel-processing operators may be initialized to a current document. The 1st processing operator may be object optimization processing. The 2nd operator may be the conventional cutting tool map / bit map processor. An object optimization processor can be continued until the 1st defect page is discovered. At this point, the 2nd processor starts actuation, without generating image data to the last independent page before a defect page is discovered. Although image data is generated from this point to a defect page, it is not outputted to IOT170. Next, to the independent defect page after first page, it is generated and a defect page and all the continuing pages are outputted to IOT by the conventional processor. If it meets with the independent page of the beginning after a defect page, an object optimization processor will start analysis of a graphical operator, without outputting print data to IOT from a defect page to the independent page of the beginning after a defect page.

[0118] From this point, an object optimization processor generates the data by which object optimization was carried out again, and outputs it to IOT170, and it is continued until it meets with the following defective page. At this event, the graphical condition of the conventional processor is updated from the 1st independent page after the 1st defective page to the independent after [the 2nd defective page] last page. Next, above-mentioned processing is repeated until the page of the last of a current document is printed.

[0119] However, if it determines that step S220 has an enough memory resource, control will continue to step S240 and a scan line and a command will be generated [then,] with the scan line base. Next, in step S250, real-time channel data is extracted and it is stored in memory 150.

[0120] It is determined whether steps S230 and S250 continue to step S260, and have a page [document / current] further to be processed there. Supposing it is, control will return to step S200. However, if there is no page processed further, control will continue to step S270 and will return control to step S40.

[0121] Drawing 11 shows the processing for joining together using object optimization decomposition and the rendering of step S70 of drawing 9, and printing more to details. As shown in drawing 11, the processing which step S70 combines and is printed is started at step S300. In step S300, the compressed data to the next page (the 1st) of the set (namely, page of a current document) with which the page was collated is obtained. The data with which this following page was compressed is obtained from the RAM portion 151 of memory 150 with a bus interface 1610 through a bus 114.

[0122] Moreover, the compressed data may be stored in the nonvolatile memory portion 152 of memory 150. This is useful when it desires a reprint [without the need of repeating processing for a creator preparing print data using object assignment compression and the rendering of step S30 / the copy of a document page]. In order that the compressed data may control an object rendering function, thereby, object optimization capacity is not lost including meta-bit data.

[0123] Next, in step S310, using the multi-channel combiner 1630, compression discharge is carried out and the data with which the current page was compressed is combined. Next, in step S330, in order to optimize the page printed by IOT170, the meta-bit information offered to the IOT controller 160 is used so that the object optimization rendering method may be chosen and it may choose between simultaneous input streams.

[0124] That is, the multi-channel combiner 1630 is the object base, and in order to determine parameters, such as color space conversion, a tone playback curve, halftone generator frequency and a screen angle, the maximum laser power set point, and/or other information, it can use meta-bit information. The optimal data is determined based on processing of a different type applied to data under control of a meta-bit, and is outputted to IOT170.

[0125] Moreover, they may be used by the multi-channel combiner 1630 so that other meta-bit information may be outputted to IOT170 by the multi-channel combiner 1630, while some of meta-bit information generates object optimization data. In order that these meta-bits may be

outputted to the IOT170 control subsystem of IOT170 and these meta-bits may optimize print data further, color space conversion, a tone playback curve, the maximum laser power set point and/or halftone generator frequency, and a screen angle are included.

[0126] Moreover, all the meta-bit information is transmitted to the subsystem of IOT170 with print data by the IOT controller 160. Furthermore, some modes of object optimization renderings, such as the maximum laser power set point, color conversion, and tone playback, may be applied between the processings for preparing print data using the object assignment rendering of step S30, and compression while a meta-bit controls other modes in IOT170 in the multi-channel combiner 1630.

[0127] If a current page is printed by IOT170 in object optimization form in step S330, control will progress to step S340 and it will be determined whether the page of the last of a current copy was printed there. When the last page is not printed yet, control returns to step S300. However, if the print of the page of the last of a current copy is completed, control will continue to step S350 and it will be determined whether all the pages of hope were printed there. Supposing that is not right, it will progress to step S360, and the increment only of 1 is carried out, then the number of copies [there] progresses to step S370, the pointer of a current page will be reset to the page of the beginning of a current document, and, finally control will return to step S300. However, if step S350 determines that the last page was printed, control will continue to step S380 and will return control to step S90.

[0128] Drawing 12 shows the processing for constituting the object list which has the object optimization rendering tag of step S210 of drawing 10 more to details. As shown in drawing 12, the configuration of an object list is started by reading a PDL document file in step S400. Next, in step S410, the language construct of the following (beginning) is obtained as a current language construct, and is analyzed. After a current language construct is analyzed, a standard graphics interface is called. The analysis of PDL led to a standard graphics interface and other actuation changes from one PDL to other PDL. These processings are well-known.

[0129] Next, it sets to step S420, and the present language construct is analyzed in order that it may determine whether direct the present end-of-page condition. If a current language construct denotes a current end-of-page condition, control will progress to step S430 and the clipper field of a current page will be verified. Clipper verification processing is further stated to details in relation to step S1140-S1160 of drawing 19. If a current clipper field is verified in step S430, control will progress to step S440 and control will be returned to step S220.

[0130] If a current language construct does not have a finger example of a end-of-page condition current at step S420, control will progress to step S450. In step S450, a current language construct is checked in order to know whether it is a color operator. Supposing it meets, control will progress to step S460 and color operator processing will be performed.

[0131] However, in step S450, if it is determined that a current language construct is not a color operator, control progresses to step S470, and a current language construct will be checked in order to know whether it is a masking operator. If it becomes so, control will progress to step S480 and masking operator processing will be performed.

[0132] However, in step S470, if a current language construct is not a masking operator, control progresses to step S490, and the following language construct will be checked in order to know whether it is a condition operator. If it becomes so, control will progress to step S500 and graphical condition operator processing will be performed. All steps S460, S489, and S500 return to step S410.

[0133] If it is finally determined that a current language construct is not a condition operator in step S490, control progresses to step S510, and error indication will be outputted by the object optimization ESS 100 in order to direct that current language must have been analyzed appropriately. Next, control returns from step S510 to step S20 through step S520.

[0134] It should be understood that the color operator checked at step S450 contains "setcolor", "image", "colorimage", and a operator like the command of equivalence. Of course, it depends for a actual command on PDL language. Similarly, the masking operator command checked at step S470 contains "fill", "stroke", "character", and such other commands. Moreover, it depends for a actual command on PDL language. Finally, the condition operator

command checked at step S490 contains "setclipper", "setstrokewidth", "setrenderhint", and a command like such other operators. Moreover, it should be understood that it is dependent on the PDL language with which a actual command is used.

[0135] Drawing 13 shows the processing for generating the scan line data of step S240 of drawing 10 more to details. As shown in drawing 13, it starts by initializing an active object list at the processing step S600 for generating scan line data, and setting a scan line to 1.

[0136] Next, in step S610, it merges to the active object list with which each new shelf object from a current scan line aligned. Drawing 28 shows the general-purpose form of the scan line object list data generated during the configuration of the object list which has an object assignment tag in step S210. As shown in drawing 28, the pointer which points out the object started on the 3rd scan line 1513 points out the 1st object (beginning) 15131 on the 3rd scan line 1513, and, on the other hand, it points out the 2nd object 15132 on a scan line 1513. In this structure, an object list as shown in "15131" and "15132" contains the object positioned by the list according to the relative ranking of the corresponding PDL language construct which generates them.

[0137] however, the active (it sorted) object which aligned -- the starting position of an object -- and it aligns in right sequence from the left. Thus, the 1st object of the active object list which aligned is an active object which has a leftmost starting position. Similarly, the object of the last on each scan line is an object which has ** with a rightmost starting position. Scan line ***** each of that object is merged to an active list by processing of a given scan line so that sequencing to the right from this left may be maintained. Generally, an active object list starts in the scan line before a current scan line, and contains the object which is still active.

[0138] If it merges into the active object list with which a new object aligned at step S610, control will progress to step S620. In step S620, it is generated from the active object list with which the run list to a current scan line aligned. next, determining which run of a run list has a run list to a there current scan line in a "crowning" along with each point of a scan line by control progressing to step S630, i.e., the point, -- all -- others -- it graduates by determining the run which is not in the bottom of a run (flattened).

[0139] In step S630, smoothing of a run list advances control to step S640. In step S640, the sequencing list of the commands and colors to a current scan line is generated. A command is generated in order to ensure being set up appropriately so that that a suitable color can use with a color register, DISUIBURU [mask data / enable or] appropriately, the sampled image data being able to use if needed, and a meta-bit may optimize hardware processing of the laser modulation set point performed by compression discharge, color space conversion, tone playback curve processing, the IOT controller 160, and IOT170. The color in the form of referring to the pallet is generated in order to ensure that reading appearance of an exact color and the rendering tag information is carried out from the pallet to each instruction.

[0140] Control progresses to step S650 after step S640, and it is removed from the active object list with which the item consumed there aligned. An object is consumed when a current scan line is a scan line of the last in which an object appears. Next, in step S660, the number of scan lines is checked in order to determine whether the current number of scan lines is the scan line of the last of a page. Supposing that is not right, control will progress to step S670 and the increment of the number of scan lines will be carried out only for 1 there. Flows of control return from step S670 to step S610. However, if a current scan line is the last scan line, control will be returned to step S250 through step S690.

[0141] After scan line data is generated at step S240, control progresses to step S250. Drawing 14 shows step S250 more to details. Processing is started at step S700 by extracting, compressing and storing command data. the feature of command data sake -- Lempel -- a jib -- it is used in order that well-known compression technology like -- Welch (LZW) may compress command data. Command data is extracted from a smoothing run with the scan line base. Command data is stored in a part for the command channel data division of the RAM portion 151 as shown in drawing 31.

[0142] Next, in step S710, fixed color data are extracted, compressed and stored so that it may be explained in full detail below. Next, it sets to step S720, and meta-bit data are extracted,

compressed and stored. It sets to step S730, a mask pointer is extracted and stored, and the mask data which those mask pointers point out are extracted, compressed and stored. Like [in the case of command data], in order that a Prior art like LZW may compress a fixed color and meta-bit data, it is used. Like [in the case of an above-mentioned step], fixed color-data and meta-bit data and a mask pointer are extracted from a smoothing run list, are the scan line base and are stored in the fixed color channel portion of the RAM portion 151, a meta-bit channel portion, and a mask-data channel portion, respectively. These mask data are extracted similarly and then are compressed using known 1-bit compression technology like run length encoding, CCITTGroup4, and other compatible systems. Once it is compressed, mask data are stored in the mask-data channel portion of the RAM portion 151 again.

[0143] Next, in step S740, it is stored and an extract and the sampled image data are stored in the sample data channel portions of an extract, compression, and the RAM portion 151 for the sampled image pointer. Step S710- Processing of S740 is dramatically similar. However, Joint of International Organization for Standardization (International Standards Organization) Photographic Expert Although Group (JPEG) defined, different compression technology for every type [like] of data is used. Step S710- Once all parts for the data division which are different by S740, and types are extracted, compressed and (probably) stored, control will return to step S260 through step S750.

[0144] Drawing 15 shows how to process the color operator of step S460 of drawing 12 , more to details. Color operator processing is started by determining whether offer the image with which the color operator was sampled at step S800. If it is not the image with which it was sampled, control will progress to step S810. In step S810, a color palette is checked, in order that the fixed color directed with a current language construct may determine whether it already has the same entry as a color palette. If a pallet entry is established first, the rendering tag field will be set to a default. An answer is affirmation, when it checks in order to know whether there is any same pallet entry as long as the rendering tag leaves a default. If it becomes so, control will progress to step S820 and the pallet entry which has the same color model as being directed with the current following language construct there and pixel data will be detected.

[0145] However, for example, if the rendering tag is changed next, an answer is negation and a new pallet entry needs to be generated by step S1050 of drawing 16 . Thus, control progresses that the answer of step S810 is negation to step S830, a new pallet entry uses the color model and pixel data which are directed with a current language construct by ****, and it is established in a color palette. Next, in step S840, in order that a hashing function (function) may determine a suitable pallet index, it is applied to the pixel data directed with a current language construct. As shown in drawing 30 , a pallet consists of two or more 1521 to 152 n index slots. Each hash table (table) of a pallet 1520 has many pallet entries, and each of a pallet entry may be in any of a fixed color entry and the sampled entry. A pallet is stated more to details later. Next, in step S850, a current pallet entry is inserted in a pallet by the hash table slot determined by the pallet index.

[0146] Instead of the setup of a new pallet entry, a current fixed color is independently storable until it needs for step S1050 to set a rendering tag to this current fixed color. In this case, it is formed by the rendering tag field by which a new pallet entry is already reset from the default only at this event.

[0147] However, step's S's800 decision of that the color operator offered the sample image advances control to step S860. At step S860, an image resolution divisor is calculated based on image data and the IOT resolution which can be used. As stated with reference to the bits 6 and 7 of a table 4, since the resolution which can be obtained to IOT is generated, an image resolution divisor directs what clock cycle ***** of each pixel of the sampled image is carried out.

[0148] next, the latest "the current conversion (currenttransform)" evaluated by the image resolution divisor and graphical condition operator processing in which current image data was determined at step S860, in step S870 -- following -- a revolution -- and/or, a scaling is carried out. Next, in step S880, a new color palette entry uses the image data by which a current color model and current were sampled, and is established.

[0149] Next, in step S890, a pallet index is calculated by applying a hashing function to image data height (S size), i.e., "S-Size." Next, in step S900, a current pallet entry is inserted in a color palette by the hashing table slot determined by the pallet index.

[0150] Next, from either of steps S820, S850, and S900, control progresses to step S910 and the "currentcolor (current color)" pointer generated in graphical condition operator processing there is set to the current pallet entry determined by steps S820 and S850 or S900. Next, control progresses to step S410 through step S920.

[0151] Drawing 16 shows how to process the masking operator of step S480 of drawing 12, more to details. Masking operator processing is started at step S1000 by determining whether parameter "renderhint(rendering hint)" is already set by graphical condition operator processing. Supposing that is not right, control will progress to step S1010 and an object optimization rendering tag will be determined automatically there. However, if parameter "renderhint" is set, control will progress to step S1020 and an object optimization rendering tag will be drawn from a "renderhint" parameter (an unit or plurality) there.

[0152] That is, it is determined by analyzing an object type first to a current language construct depending on the number of the level of distinction offered between the types with which the object type as which, as for the object optimization rendering tag, the current language construct was determined when becoming an object to which parameter "renderhint" is not set differ from each other. If only the single level of distinction is enabling, the object optimization rendering tag determined in step S1010 will distinguish an image object (for example, halftone object) from for example, a non-image object (for example, a text and a line art object). Supposing the level of an addition of distinction is enabling, an additional rendering tag will be generated so that saturation level which is different in the same object, which color-space-changes and is different and which tone-playback-curves and is different may be offered. It is generated for every object in order to direct the laser power set point used when at least one rendering tag writes in the data of the object in any case.

[0153] A set of a "renderhint" parameter overrides the value determined automatically, otherwise, the value determined at step S1010 with an explicit instruction from a document creator in step S1020. Thus, if a document creator is not so, it can override the rendering tag set at step S1010. Therefore, although a "renderhint" parameter may not be required, it can specify what an object type is. It may be the independent type of the hint which directs the object with which a document creator wants to gain an observer's eyes irrespective of which object type this object type is. Or "renderhint" has this object in a background and you may direct not to want to gain an observer's eyes. "renderhint" may direct the thing for which the edge of an object should be made into Sharp and which do not come out. "renderhint" may direct that saturation should not be boosted and the color by which this object was defined should be held. "renderhint" may direct that a different specific laser power set point from a default set point to the type of an object should be used instead.

[0154] Moreover, "renderhint" may define an object type by the remaining portion of analysis following a default mode clearly based on the subparameter of the defined object type and the undefined.

[0155] Next, from step S1010 and step S1020, control progresses to step S1030 and the rendering tag of the "currentcolor" condition set by graphical condition operator processing by **** is updated. If a "currentcolor" tag is not in agreement with the rendering tag to the pallet entry as which it was instructed by "currentcolor" instead of a default tag as shown in drawing 12, and 15 and 16, the pallet entry must be copied, the rendering tag of the new pallet entry will be updated, and a "currentcolor" pointer will be updated by the new pallet entry.

[0156] Moreover, when the new pallet entry is not generated between the processing color operator step S460, color data are not inserted in a pallet and held separately. In this case, a new pallet entry is generated at step S1030, and that color value is set to the place held separately. This new rendering tag of a pallet entry is set to the value determined in step S1010 or step S1020. This new pallet entry is inputted into a pallet at this time by performing steps S840, S850, and S910 as a portion of step S1050.

[0157] Next, in step S1040, since one or more primitive (Lord) objects, such as a box and a bit

map, are generated, scanning conversion of the current object is carried out. Scanning conversion is well-known processing. Next, it is determined at step S1050 whether all the primitive objects were processed. If one of primitive objects remains, control will progress to step S1060 from step S1050, and the following primitive object (beginning) will be obtained from a masking operator as a current primitive object by ****. Next, in step S1070, primitive masking object processing is performed to a current primitive object. Next, control returns to step S1050. This loop continues until it determines that there is no primitive object by which step S1050 should be processed. In this case, control returns to step S410 through step S1080.

[0158] The more detailed version of the graphical condition operator processing stated at step S500 of drawing 12 is shown. In drawing 17, graphical condition operator processing of step S500 is started by determining whether the "setclipper" operator was set at step S1100.

[0159] Control will progress to step S1110 and that the "setclipper" operator is not set will determine whether the "setrenderhint" operator was set to, if it becomes. Supposing that is not right, it progresses to step S1120, and control will set a graphical condition to this condition operator, and will progress to step S1180. However, if a "setrenderhint" operator is set at step S1110, control will progress to step S1130 and the "renderhint" parameter in a graphical condition will be set to the rendering hint directed by the "setrenderhint" operator there.

Control progresses step S1180 again from step S1130.

[0160] However, in step S1100, if a "setclipper" operator is set, control will progress to step S1140, a there current clipper object will exist, and it will be determined whether the integrity attribute is "perfect." If it becomes so, control will progress to step S1150 and a current perfect clipper object will be changed into a sweep object there. Next, the sweep outline shown as a field r1 is set to a current clipping region. Simultaneously, the fill (restoration) field of a sweep shown in a field r2 is set to the sweep object under a current clipping region. However, when a current clipper object does not exist, or when the current existing clipper object does not have the integrity attribute of "complete (it is perfect)", control progresses to step S1160 directly from step S1140.

[0161] In step S1160, this object is inserted in a scan line alignment active object list as a scan line of the beginning of an object. Step S1160 forms the "clipper verification" processing S430 which was shown in drawing 12 and mentioned above from step S1140. Next, control progresses to step S1170 and the "currentclipper" command of a graphical field is set [then,] to a new clipper field. Next, in step S1120 and step S1130, control progresses to step S1180 and control returns to step S410.

[0162] As step S620 of drawing 13 was described, more detailed description of the processing for generating a scanning line list is shown in drawing 18. The processing for generating a scanning line list is started in step S1200 by setting variable "thisobject" to the object of the beginning of the active object list of a current scan line.

[0163] Next, in step S1210, variable "thisobject" is tested in order to determine whether it points out an effective object. Supposing variable "thisobject" does not point out an effective object, control will return to step S630 through step S1280. However, if "thisobject" points out an effective object, it will always become truth immediately after setting "thisobject" to the first object, and control will progress to step S1220.

[0164] In step S1220, "thisobject" is checked in order to know whether it is a sweep type object. If it becomes so, control will progress to step S1230. It merges into the run list with which one run aligned for every piece of s1 outline of "thisobject" on the present scan line in step S1230. Each of the merged run consists of a starting position in alignment with the scan line, and a termination location. Each run consists of a pointer which points out the lower layer sweep object s2 of "thisobject" again, and a layer and color data may be extracted. As mentioned above, s1 field was effectively set to the clipping region, when a sweep was generated, and another side s2 field was set to the lower layer sweep object of a current clipping region. Control progresses to step S1270, after performing step S1230.

[0165] In step S1220, when the class of "thisobject" is not a sweep, control progresses to step S1240, and "thisobject" is checked there in order to know whether it is a clipper type object. If it becomes so, control will progress to step S1250. In step S1250, from a "thisobject" inside list,

each object resists the clipper of "thisobject" and clips. Inside lists are lists of objects added to this list while being collected for this clipping object between the primitive masking object processings of step S1740-S1770 which are shown in drawing 23 and stated more to details below. That is, each inside list of objects relates to the clipping object which corresponds while the clipping object which corresponds as a current clipping region was effective. Like what was mentioned above about s1 sweep outline, the clipper of "thisobject" was set to the effective clipping region, when [this] clipper object generation was carried out. An object is clipped by removing all the portions of the object on the outside of the clipper of a clipping region.

[0166] after an object clips, each profit **** run is merged into the run list which aligned. Each run consists of a starting position in alignment with a scan line, and a termination location in step S1220. However, each run consists of a layer from the inside list of "thisobject", and a pointer to the clipped object to color data in step S1250. Like the case in step S1230, once processing is completed, control will progress to step S1270.

[0167] In step S1240, when the class of "thisobject" is not a clipper, control progresses to step S1260. In step S1260, it merges to the run list with which the run for every piece of "thisobject" of the present scan line aligned. Each run consists of a starting position in alignment with a scan line, and a termination location like [in the case of step S1230 and step S1250]. However, it consists of a pointer to "thisobject" [as opposed to /****/ in this case / again / a layer and color data in each run]. Like [in the case of step S1230 and step S1250], once processing of step S1260 is completed, control will progress to step S1270.

[0168] It is set to the object variable "thisobject" is instructed to be by the "next" field of current "thisobject" in step S1270. Therefore, following "thisobject" turns into current "thisobject". Control progresses to step S1210 from step S1270.

[0169] Drawing 19 shows the processing for graduating the run list of step S360 of drawing 13 more to details. As shown in drawing 19, the processing which graduates a run list is started at step S1300. In step S1300, the foreground and background object of a current run are initialized by the white object which has a default rendering tag. A starting position and a termination location are initialized by initiation of a current scan line. Finally, variable "currentstart" is initialized by initiation of a scan line.

[0170] Next, in step S1310, variable "currentstart" is checked in order to determine whether termination of a scan line was reached. When that is not right, control progresses to step S1320 and the next visible run is identified there.

[0171] Next, control progresses to step S1330, and it checks in order to determine whether they of the foreground and background object color, and rendering tag of the next visible run are the same as that of the foreground of the object of a current run and a background object color, and a rendering tag there. If this is truth, control will progress to step S1360 and a run will be combined there by setting the termination location of a current run to the termination location of the next visible run. However, when this is not truth, control progresses to step S1340 and the there current command and there current color to a run are generated. Next, in step S1350, the next visible run turns into a current run.

[0172] Then, both steps S1350 and S1360 progress to step S1370, and it is set in the termination location of the present run of "currentstart". Next, the control from step S1370 progresses to step S1310. In step S1310, if "currentstart" reaches to termination of a scan line, control will progress to step S1380 and will be generated there for a current run of the set of the last of a command and a color. Control returns from step S1380 to step S640 through step S1390.

[0173] Drawing 20 shows the processing for generating the command and the present present color to a run of a scan line of steps S1340 or S1380 more to details. [of drawing 19] In step S1400, the foreground object of a current run is investigated, in order that the object may determine whether to be transparence or not. That is, the enquiry is whether to be the object 15570 of the bit map type with which the object has the true clear field 15574. Supposing that is not right, control will progress to step S1410. In step S1410, the foreground object of a current run is investigated in order to know whether the object is the class sweep type object 15580. Supposing that is not right, control will progress to step S1420 and will generate the Normal

command and a color. Next, control progresses to step S1460.

[0174] If it is determined that a foreground object is the sweep class type object 15590 in step S1410, control will progress to step S1430 and a sweep command and a color will be generated. Next, control progresses to step S1460 again. However, in step S1400, control progresses to step S1440 with decision **** in case of the object of the class bit map type with which the foreground object of a current run has the true clear field, a background chain is processed, and a mask-bit map is fixed. Next, control progresses to step S1450 and generates a command and a color using a mask. Next, control progresses to step S1460 again.

[0175] This procedure is not common in the point that therefore it may be called to two steps. thus, the step S1460 -- which of steps S1340 and S1380 -- this procedure -- call appearance -- or [bottom] -- determining -- the following step S1350 with appropriate control, i.e., a step, and S1370 -- it returns. That is, if generation of a command and a color is called at step S1340 to a current run procedure, control will progress to step S1350. Similarly, if generation of a command and a color is called at step S1380 to a current run procedure, control will progress to step S1390 and control will be returned to step S640.

[0176] As mentioned above, step S1400 operates so that the object of the present run may determine a foreground color, while the object directed by that background field determines the color which is "0" bits of the mask of this run, when the foreground object of the present run is transparency (namely, bit map class object). However, you may be the transparency bit map which is the same or has a different foreground color in itself [background object]. Two adjoining clear layers may be combined by applying a logic OR function to those bit maps like [when a foreground color is the same] the case where it has a black alphabetic character bit map. Since the original writing of the mask channel map of this field was overwritten by the lower bit, OR of the bit map of a lower bit map object is carried out to that map at this step. Carrying out OR of the transparency background to the object on it, and joining together can continue until it reaches a transparency background layer using the color of different "1" bit from the upper object. At this point, the conflict in the desirable activity of a mask channel was seen. It is solved by changing into a run the group of "1" bit which adjoins spatially [the transparency background bit map which carries out a conflict], and applying "smoothing" processing recursively. Thus, the transparency background object which carries out a conflict is changed into the set of the object which is not transparent. This processing follows a background chain until an opaque object or a layer "0" white paper is discovered.

[0177] Drawing 21 shows more detailed description of the fixed color extract of step S710 of drawing 14, compression, and storing processing. An extract, compression, and storing fixed color processing are started at step S1500. In step S1500, following referring to the color (criteria) (beginning) is obtained as current refer to the color. Present referring to the color is a pointer which points out one of the pallet entries of the pallet data structure shown in drawing 30.

[0178] Next, in step S1510, color data are obtained from a current pallet entry. The color class field 15233 which directs whether it is the image pallet entry which the rendering tag field 15232 which offers the following link 15231 to which each fixed color type palette entry 15230 points out the next pallet entry of this hash table slot, and the data about what the rendering of this pallet entry should be carried out, and this are fixed color palette entries as shown in drawing 30, or was sampled, or it is the pallet entry of other types is included. The color model field 15234 directs the desirable color space conversion for changing this color into CMYK with a rendering tag field while directing which type of color model (for example, RGB, CMYK, CIE Lab, or in addition to this) pixel data uses. Finally, the pixel value field 15235 stores actual color data.

[0179] Next, in step S1520, reading appearance of the pixel data of a current pallet entry is carried out from the pixel value data field 15235. Next, in step S1530, the pixel data by which reading appearance was carried out exactly is stored in the channel data structure of RAM151 in the location which can be used for the degree of a fixed color channel, as shown in drawing 31.

[0180] It is foreknown that the pixel data which consists of mixture of a color model is stored in a channel data structure in this way. It is changed into CMYK (or other IOT assignment) data under control of a meta-bit in IOT170 or the multi-channel combiner 1630 as mentioned above.

Moreover, the color conversion for which the color model and the rendering tag field opted can be applied to pixel data, and can change it into CMYK as a part of step S1520.

[0181] Next, in step S1540, a pallet is checked in order to determine whether current referring to the color is last referring to the color. Supposing it is not so, control will be again chosen as step S1500 as current in return and its following refer to the color refer to the color. However, when current referring to the color is last referring to the color, control progresses to step S1550.

[0182] In step S1550, from memory, reading appearance of the color data in fixed color channel data channel structure is carried out, they are compressed, and are re-stored in compression form. The data transfer through a bus 114 is minimized by compressing the fixed color data stored in the fixed color channel field like [in the case of other channel data].

[0183] If fixed color data are compressed and it is stored in step S1550, control will return to step S720 through step S1560.

[0184] Drawing 22 shows more detailed description of the meta-bit extract of step S720 of drawing 14, compression, and storing processing. As shown in drawing 22, the processing which extracts, compresses and stores a meta-bit is started at step S1600.

[0185] In step S1600, following referring to the color (it being a pointer to a pallet entry)

(beginning) is obtained as present refer to the color similarly in step S1500 of drawing 21.

Similarly, in step S1610, a rendering tag is obtained from the rendering tag field 15232 of the fixed color palette entry 15230, and the sampled rendering tag field 15242 of the image pallet entry 15240 similarly in step S1510. Since both both the fixed color palette entries 15230 and sampled image pallet entries 15240 contain a rendering tag, this processing is not limited to a fixed color palette entry as in the flow chart shown in drawing 21.

[0186] If a rendering tag is obtained at step S1610, control will progress to step S1620. In step S1620, the lookup of the rendering tag is carried out in the translation table which changes assignment between a printer independent rendering tag and an IOT assignment meta-bit. Thus, lookup processing of step S1620 returns the IOT assignment meta-bit value which offers suitable hardware **/suitable for IOT170 of assignment; or software processing control based on the rendering tag of a current pallet entry.

[0187] When a meta-bit value is acquired at step S1620, control progresses to step S1630 and it is stored in the location which can be used for the degree of the meta-bit channel of a channel data structure by which the meta-bit value over the there present pallet entry is shown to drawing 31.

[0188] Next, in step S1640, current referring to the color is checked in order to know whether it is last referring to the color. If that is not right, control will be again chosen [to step S1600] as current in following refer to the color refer to the color return and there. However, when current referring to the color is last referring to the color, control progresses to step S1650.

[0189] In step S1650, the meta-bit data stored in the meta-bit channel are compressed, and it is stored in the meta-bit channel of a channel data structure. Next, control returns to step S730 through step S1660.

[0190] Drawing 23 shows more detailed description of primitive masking object processing of step S1070 of drawing 16. As shown in drawing 23, primitive masking object processing is started at step S1700. In step S1700, a current primitive masking object is checked in order to determine whether it is a bit map primitive. As mentioned above, primitive objects are a box, a bit map, etc. If it becomes so, control will progress to step S1710, it will act to a mask-bit map as the Brit of this bit map primitive there (Blitted), and this bit map primitive will overwrite the data beforehand stored in the location which acts as a Brit. "A BURITTI ring (Blitting)" is a Prior art which is called "bit level block transfer" or "BITBLT" processing, and enables modification of memory block on a cutting tool or not the word boundary but a bit boundary. Next, control progresses to step S1720 from step S1710. When a current primitive masking object is not a bit map primitive, control progresses to step S1720 from step S1700.

[0191] Next, in step S1720, a current primitive masking object is checked in order to determine whether a clipping object more restrictive than a page boundary box is effective. When such a clipping region is not active, control progresses to step S1730 and this object is added to the

scan line object list corresponding to the first scan line to this object there. That is, this object is added only to the scan line object list of a scan line with which it appears first. Next, control progresses to step S1780 from step S1730.

[0192] However, when clipping regions other than a page boundary box are effective, control progresses to step S1740 from step S1720. In step S1740, a current primitive masking object is checked in order to determine whether be the portion of an imperfect sweep, although it is existing. If it becomes so, control will progress to step S1750 from step S1740. In step S1750, a current primitive masking object is added to the sweep subitem of a current clipper object. Although it is existing, one imperfect type of a sweep is a simple primitive object like a box. When a current primitive masking object adjoins the existing primitive object and is discovered, a new sweep object is generated and a primitive object current by existing is linked to the s2 field of a sweep. This new imperfect sweep object becomes the subitem of a clipper. Next, control progresses to step S1760 and the integrity attribute of a current clipper object is updated there. The integrity attribute of a current clipper object directs whether sufficient primitive masking object was added to the sweep subitem corresponding to a current clipper object, and was thoroughly filled up with the current clipper object (fill). From step S1760, control progresses to step S1780 again.

[0193] A current primitive masking object is existing, or if it is not the portion of an imperfect sweep, it will progress to step S1770 from step S1740, and a current primitive masking object will be added to the item list of current clipper objects there. Control progresses to step S1780 from step S1770. In step S1780, control is returned to step S1050.

[0194] As for drawing 24, more detailed explanation of step S1320 of drawing 19 of the next visible run discernment processing is shown. As shown in drawing 24, the next visible run discernment processing is started at step S1800. In step S1800, variable "thisrun" is initialized to the next run which remains in the run list which aligned. Variable "currentend" is set in the end of the next run.

[0195] Control progresses to step S1810 from step S1800. In step S1810, the run list which aligned is checked in order that it may know whether it is empty, and variable "thisrun" is checked in order to know whether it will begin after variable "currentend". If these both are falses, the run which control progresses to step S1820 and is referred to by "thisrun" there will be checked in order that the object referred to by it may know whether it has a layer on the layer of the run segment directed by variable "highestrun". That is, "highestrun" points out a run segment, i.e., the portion of a run, and it has a list of a starting position, a termination location, a foreground object, and background objects chain-ized potentially. When the layer of the object referred to by "thisrun" is not on the foreground layer of the run segment referred to by "highestrun" next, the object referred to by "thisrun" is directly under the foreground of the run segment referred to by "highestrun". In this case, control progresses to step S1830 and the object relevant to "thisrun" and it is processed as a new run [directly under] there. Control progresses to step S1860 from step S1830.

[0196] However, if the layer of the object referred to by "thisrun" is on the foreground layer of "highestrun", control will progress to step S1840. In step S1840, the starting position to "highestrun" is checked, in order that it may know whether it will start behind the starting position directed by "currentstart".

[0197] When this is not truth, control progresses to step S1850, run "thisrun" is processed there, and it becomes the new highest run. Control progresses to step S1860 from step S1850. In step S1860, it is obtained from the run list with which new "thisrun" aligned. Next, control returns from step S1860 to step S1810. However, in step S1840, if "thisrun" begins after "currentstart", control will progress to step S1870 from step S1840. In step S1870, it is set so that the termination location directed by variable "currentend" may become equal to the starting position of "thisrun". Next, control progresses to step S1880 from step S1870.

[0198] Similarly, in step S1810, if it is truth any of a test they are, control will progress to step S1880. The identified run is processed in step S1880. Next, control returns from step S1880 to step S1330 through step S1890.

[0199] There is a list of active runs on a current scan line at the beginning of smooth run list

(Flatten Run List) processing. This list aligns from the left to the right based on those extreme left points. Smooth run list processing generates from the left the stream of the command which states each portion or segment of each visible run on a scan line to accuracy in right order. The very important portion of this processing is starting at the left end of a paper, and, identifying the next longest portion of a visible run on the whole. This is attained by the visible segment discernment (Identify Next Visible Segment) processing next to step S1320. The fundamental approach is simple. Since a run aligns with those starting positions, they can be added to an "active run list" and the list of runs is considered to be active with this point of meeting a scan line now. Since a run aligns with those starting positions along with a scan line, they can be added to an active run list, when it arrives at those starting positions, and when the location in alignment with a scan line moves forward across those termination locations, they may be canceled. Next, the run which has the highest target value is identified as a "crowning (on-top)". However, two key factors, an engine-performance factor, and a transparence bit map object complicate processing.

[0200] Some graphical structures are conspicuous in PDL which makes processing of this simple type difficult. An example is known as a radial sweep. A radial sweep consists of an object with the big lowest layer, and a gradual more small object on a lower layer more, and the object of all tops enters completely substantially inside [all], the object of such the bottom. a page top. each — such a sweep can have hundreds of layers. One problem has very many objects seen during an active run list near the object of a stack a crowning or near it. The gestalt of implementation of instantiation expressed here is mitigated to abbreviation completeness by introducing directly under chain-ized processing of step S1830 of drawing 24 . Directly under chain-ization removes the present highest layer from an active run list by linking to the run which covers it, as long as the run of a lower layer re-appears after the present higher run is completed when it is covered by the higher layer. When a low run is completed by higher run before, a lower run may be thoroughly canceled in that it is covered. Thus, since the object covered temporarily is chain-ized instead of being allotted to an active run list, an active run list is maintained very short. Thus, all runs of a radial sweep are in the long chain directly under a current top run. After the upper run is completed more, the run of the beginning of the chain [directly under] of the run is added to an active run list. If a new run encounters the run list which is not in a crowning and which aligned at the point, it will be added to the chain directly under a suitable layer. That is, it is added under the run which is the upper part of the run which has a layer below self, and has a higher layer. Furthermore, a run is canceled as soon as ending before being directly under [where it was removed from the chain / directly under / , and they were newly inserted] a run and making insertion is known. It means that it does not become visible [those runs] already [this] at the existing point to the right.

[0201] The 2nd complexity is caused by the transparent bit map object. "0" bits are transparence (clear), although a bit map object has the special feature of transparence and "1" bit is colored there. That is, the color of the object [directly under] of a bit map object is transparent, and visible. These both complicate chain-ization [directly under] and do the conflict of the activity of a mask channel. The latter difficulty is processed with the technology later mentioned as a portion of the discernment run processing steps S1350 and S1380 of drawing 19 . If a directly under chain is always emptied, it returns to an active run list and all the transparent runs are removed from an active run list when the difficulty of chain-izing has a transparent top run, in the gestalt of implementation of instantiation, it will be processed by reconstructing a chain [directly under] again.

[0202] Drawing 29 shows the expression with which the general-purpose object data structure 1550 stored in memory 150 was used widely. The general-purpose object structure 1550 consists of these numbers and types that change in a link field 15510, the layer field 15520, the pallet entry pointer field 15530, the object class assignment procedure field 15540 (the above-mentioned field is a fixed field), the following object data field 15550, and a following object class. Specifically, the next link field 15510 is used in order to form the list of objects by pointing out the link field 15510 of other one object as follows. Of this mechanism, the scan line object list 1510 shown in drawing 28 is formed.

[0203] The layer field 15520 is used in order to encode the relative height of an object along with the Z-axis prolonged vertically from a page. That is, the PDL file which describes a page image is constituted so that it may be covered by those objects defined after the graphical object of the page image previously defined as a PDL file filing. This object defined previously can be stated to the stack of the object which forms a page image that it is more low along with the Z-axis. It can be said that similarly the object which appears in a PDL file later is more high along with the Z-axis. The layer field 15520 encodes this relative height of the object of a stack. Each new shelf object can give the level value which becomes high continuously during primitive masking object step S1070 processing of drawing 16 .

[0204] The pallet entry pointer field 15530 is a pointer to the entry in the pallet data structure 1520 of memory 150. The pallet entry referred to may be a pallet entry generated as a result of the color operator processing step S460 which is restricted neither to fixed color data nor the sampled image data, but is shown in drawing 15 . The object class assignment procedure field 15540 is a pointer to the collection object of a procedure which changes from one object class to other object classes in detailed actuation of a procedure. Thus, the object class assignment procedure field 15540 of all the objects that have the same type or the same class shows the same procedure. The procedure of an assignment object class can access accuracy to the data of the object class assignment.

[0205] The object assignment class data field to the box list class data field 15560 shown also in drawing 29 consists of the linked box pointer field 15561 and the present box pointer field 15562. Both the linked box pointer fields 15561 point out a series of boxes which form an object. Each box of the box of this ***** consists of the link to the next box of this box of a series of, the lower left of this box, and the location of an upper right corner. Some well-known technology allots useful constraint conventionally to the box which may be expressed by such a series of boxes. Some constraint is useful although the engine performance of the clipping procedure which acts on the clipping region described by such box that stands in a row is increased. Thus, the current box pointer field 15562 of the box list class 15562 is offered as a thing convenient for clipping and other procedures.

[0206] In the bit map assignment class data field 15570, this bit map has boundary box data of that self including the bit map object pointer field 15571. The data bit which has the value of 1 in a bit map directs to say that the object in the point expressed with the bit should be printed in the color referred to by the pallet entry pointer 15530. The data bit which has the value of 0 takes one side of two semantics which carries out alternation depending on the value of the clear field 15574. In the case of 0 to which the clear field 15574 expresses a false, the data bit of 0 of a map expresses white. the clear field 15574 expresses truth -- un--- when it is 0, the data bit of 0 of a map has a transparent bit map at those points, and means that a color is determined by the object under a bit map object.

[0207] The outline object pointer field 15572 expresses the outline or the boundary of the bit map object 15570. Generally the outline object referred to is a box list class object. Thus, a complicated configuration can be expressed and bit map 15770 the very thing is able to be a rectangle. When a bit map is transparent, the background object pointer field 15573 is used between data smoothing, so that it may be directed by the clear field 15574.

[0208] The clip assignment class object data field 15580 includes the clipper object pointer 15581, the object inside clipper field pointer 15582, and the integrity attribute 15583. The clipper object pointer 15581 points out a box list class object, in order to specify the configuration of a clipper field so that it may be set by the "setclip" operator step S1170 of drawing 17 . When the clipper field expressed by this clip class object is the present clipping region, the object inside pointer 15582 is processed at the processing primitive masking object step S1770, as shown in drawing 23 . The integrity attribute field 15583 is used in order to encode by what kind of method perfect sweeps should be collected thoroughly how selectively, while the current clip class object showing a clipping region is filled up with the boundary box of a clipping region (fill).

[0209] The sweep assignment object class data field 15590 consists of s1 object pointer field 15591, s2 object pointer field 15592, the sweep change rate field 15593, and the coloring run pro juicer method field 15594. In order that s1 object pointer field 15591 may point out the outline

object of a sweep, it is mainly used, and a box list class object expresses an effective clipping region, when the objects which consist of a sweep are collected. s2 object pointer field 15592 points out the 2nd object of a class sweep, and the s1 and s2 object points out two edges of the linked list of the simple object showing the colored slice from which a sweep changes. The change rate field 15593 of a sweep is calculated, when the clip class object used in order to collect objects on the inside of a clipping region is changed into a sweep class object as are shown in drawing 17, and shown in the graphical condition operator processing step S1150. It is used in order to determine whether be the color from which a sweep changes by sufficient frequency to guarantee the activity in the "slow sampling channel" mode above-mentioned actuation.

[0210] The coloring (color) run pro juicer method 15594 is the procedure of the type proper of the collected sweep. The sweep which changes vertically, the sweep which changes horizontally, and the sweep which has other various features are different color run pro juicer method procedures. This procedure is called between the generation sweep command shown in drawing 20, and the color step S1430, and generates a command and a color for each [which this field 15594 can point out] coloring slice of every.

[0211] Drawing 25 shows the Normal command of step S1420 of drawing 20, and more detailed explanation of color generation processing. The Normal command and color-generation processing are started at step S1900, and the color class of an object is checked there in order to determine whether have the image with which the object was sampled as the color. If it is no, control will progress to step S1910 and it will secure that the suitable color data and the meta-bit value over this run are loaded to one of the color registers at this step. It is determined whether it was compared with refer to [by which referring to the pallet to the foreground of this run is held at each of a shadow register to a color / meta-bit register 0-2] the pallet, and the color referred to by the object was loaded to one of the color registers before. When this is not so, color data are loaded by the command which one of a color / the meta-bit registers 0-2 is chosen, and is generated at the following steps. This selection can use all conventional processings like the "minimum activity frequency (least recently used)" processing or other same or equivalent processings. The reference to the pallet corresponding to this run is outputted to the last in the location which can be used for the degree of a fixed color channel data structure.

[0212] Next, control progresses to step S1920 and the Normal command which has a suitable bit value over the specific color register chosen by the color register-select value A, a load color, and this color register-select value A is taken out there based on the value determined at step S1910. Furthermore, if needed, many "repeat" commands are generated in order to extend a run exceeding the 64-pixel length limit in the Normal command. From step S1920, control progresses to step S1960 and control is returned to step S1460 there.

[0213] When it is the image with which the color of an object was sampled in step S1900, control progresses to step S1930, and thereby, the meta-bit value corresponding to a sample image ensures ** loaded to the meta-bit register 3, when an image pixel is displayed. A shadow register is held at the RAM portions 151 of the whole command and the memory 150 between color palette reference generating processings, and can determine the current content of the color register by it. In this case, the value of the color register 3 is not important. Only the value of the meta-bit register 3 is used and color pixel data is supplied by the sample channel. When the pre value of a meta-bit register needs to be loaded based on the content of the shadow register, the pallet entry which referring to the color palette is inserted in the degree of a fixed color channel in an available location, and has an exact rendering tag and the color value which is not not much important is referred to.

[0214] After the value of the meta-bit register 3 is verified at step S1930, control progresses to step S1940, a command displays sample channel data there, a suitable sample divisor is set up, and the meta-bit register determined at step S1930 is loaded to the location which can be used for the degree of a channel memory portion. As in step S1920, if needed, a "repeat" command is generated so that run length may be extended exceeding a 64-pixel limit of the Normal command.

[0215] Next, in step S1940, control progresses to step S1950 and the address and length of sample image data which are supplied to the command generated exactly there are loaded to the next available location of a sample pointer channel data structure. In order that the extract of step S740 of drawing 14 , storing sample image pointer processing and an extract, compression, and storing sampling image data processing may discover and extract the required portion of a sample image by the DMA controller in addition to using the address and length, the address and length are used.

[0216] Next, control returns from step S1950 to step S1460 through step S1960.

[0217] Drawing 26 shows the sweep command of step S1430 of drawing 20 , and more detailed description of color generation processing. A sweep command and color instruction generation processing start at step S2000, and the change rate of a sweep is checked there in order to know whether it is 2 pixels or less per color step. In order to generate a actual command and a actual color, the sweep color RAMPURO juicer (generation) method found by the sweep the data of an object data structure is called. This method changes with types of a sweep, extracts the background slice of a sweep, and offers initiation/termination location for every subrun in a pallet entry and a sweep object. For example, the sweep which changes from the scan line whose color is one only to the following scan line has the color run pro juicer method that find one suitable slice to a **** scan line, and only one ** notifies it to a command generating processor. Next, a command generating processor generates "a command and refer to the color to the whole run length." By contrast, the sweep which carries out a step from a color to a color along with a scan line must have "one command and one refer to the color" for every slice of an object along with a scan line. The change rate of a sweep will be calculated between clipper verification processings, once the sweep is changed from one clipper object. This field directs the number of averages of the pixel used for every color step in a sweep. When a change rate is 2 pixels or less per color step, control progresses to step S2020 and one single command which chooses a color / meta-bit register 3 using a color register-select B value is generated. As mentioned above, when using the color / meta-bit register 3 which has a color register-select B value, one color is read from a fixed color channel for every clock cycle. By using this slow sample channel, only a single command is required with refer to [of one **] the color palette for every color slice. A command bit and referring to the color palette are outputted to a location available to the degree of command channel memory and fixed color channel memory. [which set up an IOT controller]

[0218] Next, control progresses to step S1460 through step S2030 from step S2020.

[0219] However, if the change rate of a sweep exceeds 2 pixels, control will progress to step S2010. In step S2010, a command and referring to the color palette are generated, and it must be loaded to command channel memory and fixed color channel memory. Control progresses to step S1460 through step S2030 like [in the case of step S2020] from step S2010.

[0220] Drawing 27 shows more detailed description of the command which uses mask processing of step S1450, and color generation. In order to establish the conditions which enable a combiner to run this processing to Normal (activation), it is determined whether there is any special case where the command of length 1 must be published. Such conditions include the need for loading both a meta-bit value and a fixed color, when carrying out the switch between the foreground of an image, and a fixed color foreground using the need for loading the color of both transparence sweeps, or a mask. [in some of these cases], it must be investigated in order to determine the sequence that the bit color register of the beginning (the 1st) of the mask data itself is loaded, and the color chosen by it by the pixel of the beginning of mask data chooses the color register loaded by the command which takes out the pixel. The 2nd command is taken out, and the first command has a length of 1 pixel so that other required color / meta-bit registers can be loaded to the 2nd pixel clock. In some cases, mask data need for it to be reversed and to be processed. It is generated in order to choose a fixed color on an image using a mask. Since a sample channel is chosen by the color register-select A value, the mask selection between the SrcB fields in SrcB and a command must be specified in order to allot the fixed color which must be chosen by the color register-select B value of this example in this case to a foreground.

[0221] The processing is started in step S2100, and a foreground and a background color are

checked there in order to know whether it is the image color by which those any were sampled. In affirmation, control progresses to step S2110 and an image meta-bit and a fixed color are verified there for the mask. That is, the load sequence of a color register and a meta-bit is determined in order that the processing may ensure operating to accuracy. This decision is made by asking the initial bit of mask data by determining whether it determines whether there is any fixed color needed for any of a color register they are, or there is nothing, and the meta-bit value of the meta-bit register 3 needs to be loaded clearly. Next, since a command is generated at the continuing step, the number of the 1-pixel commands determined are needed here since hardware is initialized suitable for a suitable condition is used.

[0222] Next, control progresses to step S2120 from step S2110, and the data determined at step S2110 is generated actually.

[0223] Next, control returns from step S2120 to step S1460 through step S2190.

[0224] However, if it is not the image color by which the foreground and the background were sampled in step S2100, control progresses to step S2130, and a shadow register will be checked there in order to determine whether both a foreground and a background color need to be loaded. If it becomes so, it will be determined which should be first loaded among two colors in which the need of progressing to step S2140 and being loaded from step S2130 has control.

Moreover, this is made by investigating the bit of the beginning of mask data as mentioned above. When the color which needs to be loaded first is determined at step S2140, control progresses to step S2150 and a 1-pixel length command is generated there, and since the first color is loaded, it is loaded to the location which can be used for the beginning of command channel memory. Of course, the first color is loaded to the location which can be used for the degree of a fixed color channel.

[0225] Next, control progresses to step S2170 and it is loaded by generating the command to the remainder of the run length who has the mask which the color determined are loaded to the 2nd there chooses appropriately between the 2 color. Thus, the 2nd command is loaded to the location which can be used for the degree of command channel memory, and 2nd referring to the pallet is loaded to the location which can be used for the degree of a fixed color channel.

[0226] However, when both a foreground and a background color do not need to be loaded at step S2130, control progresses to step S2160, and a shadow register is checked there in order to determine whether either a foreground or a background color needs to be loaded. If it becomes so, control will progress to step S2170. If that is not right, control progresses to step S2180, a command is generated there, and it is loaded to the location which can be used for the degree of command channel memory. However, since both colors are already loaded to the register of an IOT controller, reference of an addition on a pallet is generated and it does not need to be loaded to fixed color channel memory. Next, control progresses to step S1460 through step S2190 from step S2180.

[0227] Drawing 31 shows symbolic drawing of the RAM portion 151 of memory 150. The RAM portion 151 contains a color palette 1530, the mask-bit map 1560, a pallet 1520, the scan line pointer stack 1510, the command channel memory 153, the fixed color channel memory 154, the sample color channel 155, the mask channel 156, and the meta-bit channel 157.

[0228] Drawing 32 shows symbolic drawing of the whole object optimization system which has the IOT controller 160 integrated by one system, the decomposition system 130, and a command instruction and data generation system, and has the data flow and procedure which were able to be set in order from initiation to termination. An available object optimization resource is also shown in the IOT controller 160, IOT170, and/or the decomposition system 130.

[0229] Although it was explained with the gestalt of the specific instantiation-operation in which the utility of this invention includes the system and method of this invention as outlined upwards, a clear thing has many substitution, modification, and a variation clear to this contractor. Therefore, as mentioned above, the structure used with the system and method of this invention is instantiation, and does not mean limiting this invention. Various modification does by the pneuma of this invention, and within the limits.

[0230] For example, an image may be processed by segmenting the image to a bigger segment, i.e., a field, than the size of a pixel. It can be based on how many there is how many the set point

to the laser power used in order to expose all the specific fields of an image has a halftone object or its portion in a given field configuration, i.e., the field, a text object and a line art object, or such an object. Therefore, a different laser power set point may be used in order to expose a field based on the ratio of object of different type, i.e., halftone object, text, and/or line art **. For example, a low set point can be used with the field only containing a halftone object, a middle set point can be used with the field containing a halftone object and a text, and/or a line art object, and a high laser power set point can be used with the field only containing a text and/or a line art.

[0231] that these fields are well-known distinguishable from each other, or the type of each field which uses a segment bit based on the segment algorithm developed in the future and the space coordinates of each field -- or other **** -- or it may be identified by the method developed in the future. For example, a print system user can input data at a terminal through the interface for drawing the field which should be printed using various modes of operation based on the laser power set point used when forming these fields.

[0232] Although the structure and the device which perform a laser modulation based on the difference between a halftone field, a text, and/or a line art are not necessarily limited, they are contained in all the digital copiers that modulate the output reinforcement of the power of for example, a printer, a facsimile machine and a leather raster output scanner, an LED image bar scanner, or a luminescence device and that use well-known or the device developed in the future, for example.

[0233] This invention can be carried out with super-high-degree-of-accuracy printing which uses the method of carrying out the rendering of the different halftone image from a traditional method. Super-high-degree-of-accuracy printing is performed by carrying out the rendering of the halftone dot using a different configuration. United States Patent 5485289th given to the curry (Curry) is indicating in the details of super-high-degree-of-accuracy printing. The traditional half toning method which uses a "threshold array" may also be used.

[0234] Although the utility of this invention was finally mentioned above using the various instantiation structures which generate a print image using laser, the system and method by this invention may be used with an ink jet printer like it. Therefore, a different set point as compared with the amount of ink used when not using a laser modulation, generating a halftone field and the amount control of ink generates a text and/or a line art, in order to change the size of exposure SUPPOTO is used inevitably.

[Translation done.]

* NOTICES *

Japan Patent Office is not responsible for any damages caused by the use of this translation.

1.This document has been translated by computer. So the translation may not reflect the original precisely.

2.**** shows the word which can not be translated.

3.In the drawings, any words are not translated.

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]

[Drawing 1] The block diagram of an object optimization electronic subsystem is shown.

[Drawing 2] The block diagram of the gestalt of operation of the 1st of an IOT controller is shown.

[Drawing 3] The gestalt of operation of the 1st of multi-channel KOMBANA is shown.

[Drawing 4] The gestalt of operation of the 1st of FIFO control, and a command / mask channel processor is shown.

[Drawing 5] The gestalt of the 1st desirable operation of the bus interface of an IOT controller is shown.

[Drawing 6] The gestalt of operation of the 2nd of an IOT controller is shown.

[Drawing 7] The gestalt of operation of the 3rd of an IOT controller is shown.

[Drawing 8] The gestalt of operation of the 4th of an IOT controller is shown.

[Drawing 9] Flow drawing of the whole object optimization processing method is shown.

[Drawing 10] Flow drawing for preparing print data using an object optimization rendering and compression is shown.

[Drawing 11] Flow drawing for joining together and printing using object optimization compression discharge and a rendering is shown.

[Drawing 12] Flow drawing for constituting the object list which has an object optimization rendering tag is shown.

[Drawing 13] Flow drawing for generating scan line data is shown.

[Drawing 14] Loading and flow drawing for compressing are shown for real-time data.

[Drawing 15] Flow drawing for processing a color operator is shown.

[Drawing 16] Flow drawing for processing a masking operator is shown.

[Drawing 17] Flow drawing for processing a graphical condition operator is shown.

[Drawing 18] Flow drawing for generating a scan line run list is shown.

[Drawing 19] Flow drawing for carrying out flattening of the run list is shown.

[Drawing 20] Flow drawing for generating a command and a color for a current run is shown.

[Drawing 21] Flow drawing for extracting, compressing and storing fixed color data is shown.

[Drawing 22] Flow drawing for extracting, compressing and storing metadata is shown.

[Drawing 23] Flow drawing for processing a basic masking object is shown.

[Drawing 24] Flow drawing for identifying the next visible run is shown.

[Drawing 25] Flow drawing for generating a usual command and a usual color is shown.

[Drawing 26] Flow drawing for generating a sweep command and a color is shown.

[Drawing 27] Flow drawing for generating a command and a color using mask data is shown.

[Drawing 28] Drawing of the data structure stored in memory 150 is shown.

[Drawing 29] The general-purpose structure for every object on a scan line is shown.

[Drawing 30] The general-purpose structure over a color palette is shown.

[Drawing 31] A general-purpose channel data structure is shown.

[Drawing 32] System flow drawing and resource drawing are shown.

[Drawing 33] System flow drawing and resource drawing are shown.

[Drawing 34] The block diagram of the gestalt of the 1st desirable operation of the image

processing system in IOT is shown.

[Drawing 35] The block diagram of the gestalt of operation of the 1st of object optimization print measurement and an accommodation system is shown.

[Translation done.]

特許2000-153640
(P2000-153640A)

(43) 公開日 平成12年6月6日 (2000.6.6)

(51) IntCl. ⁷	識別記号	P I	ページ (参考)
B 41 J 2/52		B 41 J 3/00	A
H 04 N 1/23	1 0 3	H 04 N 1/23	1 0 3 Z

審査請求 未請求 請求項の数 3 O L (全 67 頁)

(21) 出願番号	特願平11-327652	(71) 出願人	59000798
(22) 出願日	平成11年11月18日 (1999.11.18)	ゼロックス コーポレーション XEROX CORPORATION	
(31) 優先権主張番号	1 9 5 1 6 5	アメリカ合衆国 08904-1800 コネティ	
(32) 優先日	平成10年11月18日 (1998.11.18)	カット州・スタンフォード・ロング リッ チ ロー・800	
(33) 優先権主張国	米国 (US)	ダグラス エス. カリー	
		アメリカ合衆国 94025 カリフォルニア 州 メンロ パーク レランド アベニ ー 221	
		(74) 代理人	10079049 弁理士 中島 淳 (外 1 名)

(54) [発明の名称] プリント方法及びプリンタコントローラ

(57) [要約]

【課題】 PDL等を使用して生成されたページ画像に各
タイプのオブジェクトを形成するために使用されるレー
ザパワーが最適に制御又は選択されるようにプリントデ
ータ及びプリンタ制御コマンドを使用するプリンタコン
トローラ装置及び方法を提供すること
【解決手段】 プリント方法は、異なるタイプのデー
タを含む画像データのバイト間を区別するステップと、第
1のステップポイントでレーザパワー駆動信号を使用し
て第1のタイプのバイト画像データをプリントするステ
ップと、第2のステップポイントで前記レーザパワー駆
動信号を使用して第2のタイプのバイトの画像データを
プリントするステップと、を備え、前記第1のステップ
ポイントが前記第2のステップポイントと異なってい
る。

(2)

特開2000-153640

2

【特許請求の範囲】
【請求項1】 異なるタイプのデータを含む画像データ
のバイト間を区別するステップと、
第1のステップポイントでレーザパワー駆動信号を使用
して第1のタイプのバイト画像データをプリントするス
テップと、
第2のステップポイントで前記レーザパワー駆動信号を
使用して第2のタイプのバイトの画像データをプリント
するステップと、を備え、
前記第1のステップポイントが前記第2のステップポイ
ントと異なっている、プリント方法。

【請求項2】 バイトの画像データに対応するオブジェ
クトのタイプに基づいてそのバイトの画像データをプリ
ントするために使用される駆動信号セットポイントを調
節するプリンタコントローラであって、
プリントデータ及びメタビットデータを受信する画像出
力端末を備え、前記画像出力端末が前記画像出力端末へ
送信されるメタビットデータにより選択的に制御される
オブジェクト最適化画像形成制御サブシステムを含む画
像処理システムを有し、
前記メタビットは、前記バイトの画像に対応するオブジ
ェクトのタイプにより異なる駆動信号セットポイント間
で選択するように前記オブジェクト最適化画像形成制御
サブシステムを制御し、前記駆動信号セットポイントが
前記バイトの画像データに対応する前記オブジェクトの
前記形成に影響を与える、プリンタコントローラ。

【請求項3】 画像形成デバイスへ組み込まれ、前記画
像形成デバイスは、フックシミュレーション、プリンタ、
デジタル複写機、及びラスタ出力スキャナから成る
グループから選択される、請求項2に記載のプリンタコ
ントローラ。

【発明の詳細な説明】
【0001】
【発明の属する技術分野】 本発明は、プリント装置にお
いて、最大又は飽和レーザ又は飽和パワーを調節するた
めの方法及び装置に関する。より詳細には、本発明は、
現在露光されている画像オブジェクトのタイプに基づい
て、露光レベルを制御するための方法に関する。

【従来の技術】 デジタルカラーワークステーション、
複写機及びプリンタの発展に伴って、ページ画像を合成
しプリントするためにグラフィックアークストに以前より
依存していたページ画像のクリエータ (作成者) は、代
わりに、デジタルカラー複写機/プリンタに接続され
たコンピュータを使用して、自分自身でページ画像を生
成し、合成し、プリントする事ができる。
【0003】 しかしながら、ページ画像を生成するた
め、ページ画像をプリントエンジン命令に分解するた
め、及びプリンタエンジンを制御するためのこのような
従来技術のデジタルシステムは、ページ画像を単一で

【0004】 大きな一定のカラー部分のために最適化さ
れたページ画像において、プリンタの不安定性を隠すた
めに特に設計されたハーフトーンスクリーンは、高品質
で、テキストが無く、アーチファクトの無い一定のカラ
ー領域を生成する。しかしながら、テキストのシャープ
ネスが低下され、各ティント毎のグレイ値は、あまり調
連せず、サンプリングされたカラー部分とスライブ部
分は、許容できない。そのグレイレベルは、各ドットレ
ベルが他のレベルと関係なく別々に設計されるので、一
つのレベルから次のレベルへ滑らかにステップしな
い。

【0005】 サンプリングされたカラー部分とスライブ
部分に対して最適化されたページ画像において、スライ
ブ部分とサンプリングされたカラー部分は、低周波ハー
フトーンスクリーンにより多く利用できるグレイレベル
で利用されるので、より高い品質を示す。しかしなが
ら、テキストは、低品質で表され、且つ一定のカラー部
分は、明確な模様化を示す。
【0006】 しかしながら、各画像を単一のビットマッ
プ画像又はビットマップ画像として処理した従来のシス
テムにおいては、あらゆる一つのタイプのオブジェク
トに対して画像を最適化することは、他のタイプのオブジ
ェクトの画像品質と妥協することが必要であった。従
って、手作業による合成グラフィックアークスト分野にお
けると同様に、ページ画像を生成し、分解し、個別のオブ
ジェクトのプリント特性が最適化できるプリンタエンジ
ンへ出力すると同時に、マイクロコンピュータを使用し
てページ画像を生成する時に利用できる利点と効率を維
持するデジタルカラー複写機/プリンタ及び方法が従
来技術では必要である。

【0007】 このようなページ画像は、PostScriptTMの
ようなページ記述言語 (PDL)、InterpressTM、WindowsTM
と共に使用されるようなGraphical Display Interf
aces (GDI)、Hewlett-Packard Printer Command L
anguage (PCL-5)、等を使用して生成される。
【0008】 市販されているレーザプリンタ製品は、一
般的ではない強力なレーザパワーを使用して "白黒" 画
像を露光する事が知られている。これは、白画像と黒画
像の間でのコントラストの増強のためになされ、この通
露光による暗い領域は見る者にとって審美的に好まれ
る。この増強レーザパワーは、画像範囲の全てに提供さ

【0009】 本発明は、異なるタイプのデータを含む画像データ
のバイト間を区別するステップと、
第1のステップポイントでレーザパワー駆動信号を使用
して第1のタイプのバイト画像データをプリントするス
テップと、
第2のステップポイントで前記レーザパワー駆動信号を
使用して第2のタイプのバイトの画像データをプリント
するステップと、を備え、
前記第1のステップポイントが前記第2のステップポイ
ントと異なっている、プリント方法。
【請求項2】 バイトの画像データに対応するオブジェ
クトのタイプに基づいてそのバイトの画像データをプリ
ントするために使用される駆動信号セットポイントを調
節するプリンタコントローラであって、
プリントデータ及びメタビットデータを受信する画像出
力端末を備え、前記画像出力端末が前記画像出力端末へ
送信されるメタビットデータにより選択的に制御される
オブジェクト最適化画像形成制御サブシステムを含む画
像処理システムを有し、
前記メタビットは、前記バイトの画像に対応するオブジ
ェクトのタイプにより異なる駆動信号セットポイント間
で選択するように前記オブジェクト最適化画像形成制御
サブシステムを制御し、前記駆動信号セットポイントが
前記バイトの画像データに対応する前記オブジェクトの
前記形成に影響を与える、プリンタコントローラ。
【請求項3】 画像形成デバイスへ組み込まれ、前記画
像形成デバイスは、フックシミュレーション、プリンタ、
デジタル複写機、及びラスタ出力スキャナから成る
グループから選択される、請求項2に記載のプリンタコ
ントローラ。
【発明の詳細な説明】
【0001】
【発明の属する技術分野】 本発明は、プリント装置にお
いて、最大又は飽和レーザ又は飽和パワーを調節するた
めの方法及び装置に関する。より詳細には、本発明は、
現在露光されている画像オブジェクトのタイプに基づい
て、露光レベルを制御するための方法に関する。

50

れる。

【0009】しかしながら、この増強されたコントラストは、それに伴い、細いラインを生成することを不能にする。更に、テキスト及び/又はライニアートの透過光は、審美的に好まれるが、ハーフトーン画像の透過光は、増強されたレーザパワーがハーフトーンセルを透過光するので、不利である。ハーフトーンセルを透過光することは、ダイナミックレンジ全体に亘る制御を減少する。これは、レーザパワーの増強がプリントエンジンの全ダイナミックレンジに亘って形状開数の遷移に影響を与えるために生ずる。特に、レーザパワーが増加されると、形状開数が急早く飽和され過ぎ、ダイナミックレンジのハイ端の中遷移できない。

【0010】

【発明が解決しようとする課題】本発明は、PIL等を用いて定義されたページ画像に各タイプのオブジェクトを形成するために使用されるレーザパワーが最適に制御又は選択されるようにプリントデータ及びプリンタ制御コマンドを使用するプリントコントローラ装置及び方法を提供することを課題とする。プリントデータ及びプリンタ制御コマンドは、PILを使用して定義されたページ画像から交換されてもよいし、従来技術において公知の他のメカニズムにより提供されてもよい。

【0011】本発明は、更に、ページ画像を形成する種々のオブジェクトのオブジェクトタイプに基づいて、レーザパワー情報を含む"メタビット"情報、即ち、各バイトの画像データを最も良くレンダリングする方法についての情報を生成し、このメタビットデータを画像出力端末(107)へ送るオブジェクト最適化プリンタ制御装置及び方法を提供す。

【0012】本発明は、更に、ページ画像の各独立のオブジェクトに対するオブジェクトタイプに基づいて、自動的にレーザパワーを決定するオブジェクト最適化プリンタ制御装置及び方法を提供する。

【0013】

【課題を解決するための手段】本発明と共に使用される装置及び方法のステップの例として、PILを使用して記述され一連のPILコマンドとして格納されるページ画像は、オブジェクト最適化エレメントロニクサブシステム(0055)へ入力される。PIL分解手段は、PILで記述されたページ画像をページ画像を表すデータ構造へ分解する。このデータ構造において、独立の画像オブジェクトに関連する情報は、保持される。この情報は、オブジェクトのタイプ、及び最大レーザ変調セットポイント、カラー、最適カラー空間、層(レイヤー)情報等の他の個別の特性に関連する情報を含む。

【0014】一旦、PIL分解手段がPILで記述されたページ画像を交換すると、複写機/プリンタ、より一般的には、画像出力端末(107)、コマンド命令発生手段がページ画像のスキヤンライン毎に、データ構造を一連のコー

マンド命令、カラー指定及びメタビットレンダリング命令へ交換する。メタビットは、各オブジェクトのタイプを決定するために各オブジェクトを解析するメタビット発生手段により自動的に発生されてもよいし、ページ画像のPIL記述の生成の間にページ画像クリエータにより明示的に設定されてもよい。ページ画像の各スキヤンライン毎に、コマンド命令、カラー指定及びメタビットが発生されると、それらは、107コントローラへ出力される。

【0015】107コントローラは、分解処理の間に発生されたコマンド命令、カラー指定及びメタビットを受信する。107コントローラは、一定カラー及びサンプリングされたカラーデータを結合し、それをメタビットデータと共に107へ送信する。

【0016】107の第1の例示の実施の形態において、107は、二つ以上のハーフトーンスクリーン発生器、閾値回路、カラー空間変換回路、及びレーザパワーセクション回路を含むことができる。107コントローラから出力されるバイト幅カラーデータ及びメタビットは、107へ入力される。メタビットは、カラーデータに対してどのハーフトーン発生器又は閾値回路が使用されるか、及びどのカラー変換がカラー空間変換回路によって適用されるかを決定するために使用される。また、メタビットは、画像データをプリントするために使用されるべき択一的なレーザパワーセットポイント、即ち、最大レーザ強度、を選択するために使用される。一旦、107がカラーデータ及びメタビットからラスターデータを発生すると、レーザ強度データを含むラスターデータは、ページ画像を出力シーン上に形成するためにワーキングサブシステムへ出力する。

【0017】或いは、サンプリカルカラーデータ圧縮/圧縮解除回路、マスクデータ圧縮/圧縮解除回路、一定カラー圧縮/圧縮解除回路、及びコマンドデータ圧縮/圧縮解除回路が、バスインターフェースと統合コンパナ(結合器)の107コントローラに組み込まれてもよい。従って、これらの圧縮/圧縮解除回路は、システムバス上で107コントローラへの転送のために、データ構造の種々のタイプのデータが最適に圧縮されることができ、更に、107コントローラの第1の例示の実施の形態において、カラー空間変換器は、107コントローラへ組み込まれる。カラー空間変換器は107コントローラへ組み込むことによって、ページ画像の異なるオブジェクトに対するカラー空間変換は、丁度プリント前に実行されることとができ、更に、オブジェクトタイプに基づいて、最適化され得る。更に、カラー空間変換器は107コントローラへ組み込むことは、本発明のシステムと共に使用される107の範囲を広げる。或いは、これらの圧縮/圧縮解除回路に対する他の形態が使用されてもよい。

【0018】更に、二つのカラー空間変換器は、107へ出力されるカラーデータを発生するために使用され得

る。このように、四つのカラー層分離、C、Y、M、Kの各々を表す、4バイトのデータが同時に出力されてもよい。全4カラー層分離に対するカラーデータを同時に出力することにより、107に対するより高いページ出力が達成され、且つ107のより広い範囲が本発明のシステムと共に使用される。

【0019】本発明の具体的な形態として、第1の階層は、異なるタイプのデータを含む画像データのバイト間を区別するステップと、第1のステップポイントでレーザパワー駆動信号を使用して第1のタイプのバイト画像データをプリントするステップと、第2のステップポイントで前記レーザパワー駆動信号を使用して第2のタイプのバイトの画像データをプリントするステップと、を備え、前記第1のステップポイントが前記第2のステップポイントと異なっている、プリント方法である。

【0020】第2の形態は、バイトの画像データに対応するオブジェクトのタイプに基づいてそのバイトの画像データをプリントするために使用される駆動信号セットポイントを調節するプリンタコントローラであって、プリントデータ及びメタビットデータを受信する画像出力端末を備え、前記画像出力端末が前記画像出力端末へ送信されるメタビットデータにより選択的に制御されるオブジェクト最適化画像形成変調サブシステムを含む画像処理システムを有し、前記メタビットは、前記バイトの画像が対応するオブジェクトのタイプにより異なる駆動信号セットポイント間で選択するように前記オブジェクト最適化画像形成変調サブシステムを制御し、前記駆動信号セットポイントが前記バイトの画像データに対応する前記オブジェクトの前記形成に影響を与える、プリンタコントローラである。

【0021】第3の形態は、第2の形態において、プリンタコントローラが、画像形成デバイスへ組み込まれ、前記画像形成デバイスが、フアクシミリマシン、プリンタ、デジタル複写機、及びラスター出力スキヤナからなるグループから選択される。

【0022】

【発明の実施の形態】上述のように、ページ記述言語(PDL)で記述されたページ画像を分解するための従来の処理は、クリエータ(生成者)がページ画像全体に亘ってプリンタ依存ハーフトーンを明示的に挿入しない限り、PILを使用して記述されたページ画像を分解することによって発生されるビットマップ又はバイトマップをプリンタに対してのみ参照されるが、PILと等価な上述のあらゆるシステムが以下に述べられるオブジェクト最適化レンダリングシステム及び方法及び共に使用される。【0023】即ち、上述のように、PILを使用して記述されたページ画像を分解及びプリントするための従来のシステムにおいて、ページ画像から成る種々のオブジェクトは、PIL座標からデバイス座標へ変換され、ストロ

ークが直ぐよくされ、オブジェクトが実タスキヤンライン上のオブジェクトの位置に対応する一連のボックスへ変換され、これらのボックスは、バイトマップ(又は画像が黒/白)の場合は、ビットマップ)へロードされる。ページ画像のオブジェクトを分解することによって形成されたボックスをバイトマップへ書き込むことによつて、オブジェクトの異なるタイプ間の全ての区別が失われる。

【0024】これとは反対に、本発明において、PILにより記述されたページ画像は、ページ画像より成るオブジェクトタイプ間の区別が維持されるように分解され、ページ画像内の異なるオブジェクトのオブジェクトタイプを維持することによって、異なるオブジェクトタイプの処理は、オブジェクトのそのタイプへ最適化できる。

【0025】従来より、連続的に変化する濃度の背景を形成するために粗いスクリーンハーフトーン発生処理が利用されると同時にシャープなエッジのテキスト画像を形成するために微細なハーフトーン発生処理が利用された。オブジェクトの各タイプ毎に、最適ハーフトーン発生処理を利用することによって、背景の連続的に変化する光学的濃度が維持される。同時に、テキスト画像のシャープなエッジを維持するための最適ハーフトーン発生処理が維持される。

【0026】このように、高濃度ハーフトーンスクリーンが最適化されたテキスト画像を形成するために使用され得ると共に、色合い(tint)指定ハーフトーンセットが一定カラーの最適化されたブロックを形成するために使用され得ることが出来、更に低濃度ハーフトーンセットが最適化されサブプリングのために使用される。描写(スキャン)を形成するために使用される。

【0027】ページ画像から成る種々のオブジェクトのオブジェクトタイプを維持することによって、オブジェクトの他の特徴は、ページ記述言語により使用されるカラー空間から107で使用されるシアン(C)、マゼンタ(M)、イエロー(Y)及びブラック(B)カラー空間へ最適化される。実際に、カラー変換処理の各ステップは、アンダーカラー除去処理及びトーン(階調)再生曲線処理のように、最適化され得る。同様に、圧縮処理は、オブジェクトにより最適化されることが出来、それによって、消費される記憶リソース(資源)及び分解されたページ画像に対する遷移ロードを最小化する。

【0028】更に、要求される最適化の度合いに依存して、オブジェクトタイプの区別の異なるレベルが提供される。即ち、幾つかのインスタンス(例)において、オブジェクトが一定のカラーで着色されることを指示し、ハーフトーンスクリーン及びカラー空間変換のような全ての処理開数をこのレベルで最適化することによって十分であろう。他のインスタンスにおいて、多くの可能なカ

特開2000-153640

7

ラー空間変換の一つ又は又はは多くのハーフトーンスクリーン周波数又はは角度の一つを指定するために、追加の区別のレベルが要求されてもよく、それらの全てが一定のカラースペースに適合する。

【0029】更に、処理回路は、画像内の各オブジェクトを形成するために使用されるレーザ光露光の適切なレベルを提供するために最適化される。優れたハーフトーンドットを形成するために、形状関数は、ハイライトからミッドトーンを介してシャドウ（陰影）までの光露光のダイナミックレンジ全体に亘り一貫しなければならぬ。しかし、テキスト及び/又ははラインアートが通過光される場合と同様にハーフトーンドットの通過光に起因する問題は、ハーフトーン形成において、形状関数がダイナミックレンジのミッドトーン部分へ移動すると、そのレンジで未露光領域が露光されないままにならない。従って、得られるハーフトーンが完全なオーバーラップとなり、形状関数が全ダイナミックレンジのまだ途中にある時に、シャドウ領域に入る。従って、全ダイナミックレンジのハイエンドは、形状関数がダイナミックレンジのローエンドにある所からダイナミックレンジのハイエンドにある所までで遷移している。従って、画像のプリント品質へ影響しない。

【0030】しかしながら、テキスト及びラインアートの外観は、テキスト及び/又ははラインアート領域が通過光されると、より好ましい。理由は、テキスト及び/又ははラインアート領域がより高いパワーで露光されると、コントラストが強調されるからである。通過光により良く見えるが、あらゆる画像内に非常に細いラインのレンダリングにおける精度を失うことになる。しかしながら、失われた精度は、テキスト及び/又ははラインアートから、失われた外観に比較して大きくない。従って、テキスト及び/又ははラインアートの処理の間のレーザパワーをハーフトーンドットの処理の間のレーザパワーより大きくする事は有利である。

【0031】テキスト及びラインアート生成とハーフトーン生成との間のこれらの差は、サイズ及び制御の問題に基づく。テキスト又ははラインアートオブジェクト内のオブジェクトの個別の特徴のサイズは、ハーフトーンオブジェクト内の個別の特徴よりも大きい。従って、テキスト又ははラインアートを形成することは、ハーフトーン処理により生成される画像よりも粗い画像を生成する。テキスト又ははラインアートが通過光されると、個別の特徴があまり近接されずに離間されるので、通過光は、ハーフトーンドットが通過光される時と同程度までは形状関数を低下しない。更に、テキスト及びラインアートの形成することは、ハーフトーン形成と同様にトーンの制御を必要としない。ハーフトーン処理において、例えば、形状関数は、円形状からダイアモンド形状へ遷移し、ハーフトーンドットの画像濃度が、プリントエンジン全ダイナミックレンジに沿ってハイライト濃度から

特開2000-153640

8

ミッドトーン濃度を介してシャドウ濃度へ遷移するに従って、円形状へ戻る。フルカラー画像において、これらの形状関数によってカバーされる領域の正確な制御は、例えば、マゼンタ、イエロー及びシアンによりカバーされる領域の比率が見る事によって知覚されるカラーを制御するので、重要である。

【0032】図1に示されるように、画像出力回路（10）170によって使用できるラスターデータへ画像データのPDLフォーマットへ変換するためのオブジェクト最適化電子サブシステム（OESS）100は、好ましくはパソナルコンピュータのような汎用コンピュータ、カリフォルニア州、マリンデルのSun Microsystems 社による製造されるSunSparcTMのようなエンジェリアリングワークステーション、ミニコンピュータ等によって提供される。OESS100は、PostScriptTMやInterPressTM互換ドキュメントを発生するプログラムのようなページ画像のPDL表現を発生するためのプログラム、ページ画像のOPI表現を発生するプログラム、ページ画像のグラフィカルコマンドセットのうちのタイプは、レーザプリンタ及び/又ははインクジェットプリンタを動作するために使用されるHewlett-Packard PCL-5のコンドセットである。従って、用語“PDL”は、ページ画像のビットマップを発生する代わりに、ページ画像を記述するあらゆるタイプの表現を含むより解釈されるべきであることが理解されるべきである。

【0033】或いは、生成された直接分解されるというよりはむしろ、ページ画像のPDL表現は、不揮発性メモリやローカルネットワークやモデムを介してOESS100へ接続される遠隔地の汎用コンピュータのような何らかの遠隔PDLファイルソース手段112から受信されることとができる。従って、ページ画像を表すPDLファイルがあらゆる従来のソースから得ることができることが理解されるべきである。

【0034】ページ画像を表すPDLファイルがOESS100に一旦入力されると、それはバス114を介してメモリ150へ転送される。次に、PDLファイルは、PDL分解手段130により分解される。PDL分解手段130は、メモリ150に格納されたPDLファイルを読み出し、それを分解して図28乃至図31に示されるデータ構造を形成する。図29に示されるデータ構造は、オブジェクトリストとオブジェクトタイプを指示するレンダリングタグを含む。PDL分解手段の動作は、以下で詳細に述べられる。

【0035】PDL分解手段130が、PDL表現から発生されるページ画像の種々のオブジェクトを格納するメモリ150にデータ構造を発生すると、メモリ150に格納されたデータ構造がコマンド命令発生手段140により読み出される。コマンド命令発生手段140は、メモリ

特開2000-153640

9

150に格納されたデータ構造をスクリーンライン毎にカラーデータ、ビットマップデータ及びメタビットデータに対応する一連のコマンド命令に変換する。これらのコマンド命令、カラーデータ、ビットマップデータ及びメタビットデータが、図31に示されるように、メモリ150に格納される。

【0036】図2に示されるように、IOTコントローラ160がバスインターフェース1610を介してバス114へ接続されることが予知される。OESS100の好ましい実施の形態において、SunSPARCワークステーションが使用される。このように、バス114は、SBus（Sバス）であり、バスインターフェース1610は、SBus114と共に働くよう設計される。しかしながら、バスインターフェース1610は、従来のパーソナルコンピュータ、SunSparcTMのようなエンジェリアリングワークステーション、マイクロコンピュータ等の何れに於いても使用され得る特定のバス114と共に働くよう設計され得る。

【0037】図2に示されるように、バスインターフェース1610は、バス114に対して32ビット接続を提供する。このように、バスインターフェース1610は、各クロックサイクル間に4バイトワードを入力できる。好ましい実施の形態において、バスインターフェース1610は、連続するクロックサイクル間で16個の4バイトワードを連続的に読み出すことによつて、64バイトワードで読み出すことができる。更に、好ましい実施の形態におけるバスインターフェース1610は、バス114を介してメモリ150への直接メモリアドレス（DMA）を提供する。

【0038】バスインターフェース1610がメモリ150から4バイト部分のデータを受信すると、データは、5セットの先入れ先出し（FIFO）データレジスタのうちのセツに分配される。これらの5つのFIFOは、サンプルカラーデータFIFO1620、マスクデータカラーデータFIFO1622、一定カラーデータFIFO1624、メタビットデータFIFO1626、及びコマンド命令データFIFO1628を含む。

【0039】図2に示されるように、サンプルカラーデータFIFO1620は、FIFOの二つのバンクより成り、FIFOの各バンクが二つの512アドレスX9ビット幅FIFOより成る。このように、FIFOの各バンクの各FIFOは、バスインターフェース1610により受信された4バイトワードの1バイトを受信する。更に、バスインターフェース1610は、4個の追加のビットのデータを発生し、この追加のビットのデータのうちの、9番目のビットとして4個のFIFOの各々に格納される。これらの余分のビットの各々は、それに隣接するビットのデータが有効データであるか無効データであるかをフラグする（フラグを立てる）ために使用される。DMAデータの取り出しがワード境界のみになされるので、サンプリングされ

特開2000-153640

10

た画像に対する実際のデータがワード内で始まるということが時々起こる。この場合、実際のデータの開始に先行するワード内のバイトが無効としてフラグが立てられ、これらのバイトは、プリントされるのではなくて破棄される。

【0040】マスクデータカラーFIFO1622は、単一の256アドレスX16ビット幅FIFOデータレジスタより成る。一定カラーデータFIFO1624は、2個のFIFOの1個のバンクより成り、バンクの各FIFOは、1個の256アドレスX9ビット幅FIFOデータレジスタより成る。マスクFIFO1622及びカラーFIFO1624の各々が書き込みサイクル当たり2バイトを格納できるので、バスインターフェース1610へ入力されるデータの単一4バイト転送を利用してデータを連続的にマスクFIFO1622及び一定カラーFIFO1624の両方へ提供できる。唯一の16ビット幅内部バスがバスインターフェース1610をマスクFIFO1622、一定カラーFIFO1624及びコマンドFIFO1628へ接続し、唯一の8ビットバスがバスインターフェース1610をメタビットFIFO1626へ接続するので、2クロックサイクルが1個の4バイト転送をマスクFIFO1622、カラーFIFO1624及びコマンドFIFO1628へ書き込むために使用される。4クロックサイクルが4バイト転送をメタビットFIFO1626へ書き込むことが必要である。

【0041】メタビットFIFO1628は、単一の512アドレスX8ビット幅FIFOデータレジスタから成る。コマンド命令FIFO1628は、ペの512アドレスX8ビット幅FIFOデータレジスタの単一のバンクから成る。【0042】FIFO1620乃至1628の各出力は、マルチチャネルコンバータ（結合器）1630に接続される。マスクFIFO1622からの出力はシリアル化される。このように、唯一の単一ビット接続がマスクFIFO1622からマルチチャネルコンバータ1630へ提供される。同様に、サンプルカラーFIFO1620及び一定カラーFIFO1624は2から1（2-to-1）で多重化される。このように、サンプルカラーFIFO1620に対して、唯一の18ライン内部接続がマルチチャネルコンバータへ提供される。対の9ビット幅FIFOの各バンクは、そのデータを交互に出力する。同様に、9ビット幅接続のみが一定カラーFIFO1624とマルチチャネルコンバータ1630との間に提供される。一定カラーFIFO1624の各9ビット幅FIFOはそのデータを交互にマルチチャネルコンバータへ出力する。

【0043】FIFO1620乃至1624とは対照的に、メタビットFIFO1626とコマンド命令FIFO1628は、図2に示されるように、マルチチャネルコンバータ30へのフル幅接続が提供される。

【0044】図3に関連して以下により十分に説明されるマルチチャネルコンバータ1630は、コマンド命令FIFO1628から受信された命令及びマスクFIFO162

(7) 神岡2000-153640

11

2から受信されたビットマップデータに基づいて、サンブルカラーチャネル1620及び一定カラーチャネル1624からのデータをデータFIFO1642への単一ストリームの出力へ結合する。

【0045】データFIFO1642は、4KアドレスX9ビット幅FIFOデータレジスタから成る。マルチチャネルコンパイルがデータが8ビット幅バイトでデータFIFO1642へ出力するので、データFIFO1642の第9番目のビットが、メタビットFIFO1626から第1番目（最初）のビットを結線するのに使用される。また、マルチチャネルコンパイル1630は、出力メタビットFIFO1640へ接続される。出力メタビットFIFO1640は、4KアドレスX3ビット幅FIFOデータレジスタよりなる。マルチチャネルコンパイル1630は、各8ビットメタビットバイトを少なくとも二つの4ビット幅メタビットニブルへ分割する。各4ビットニブルの第1番目のビットが、データFIFO1642へ出力されると、各4ビットニブルの第2番目から第4番目のビットが出力メタビットFIFO1640へ出力される。実際に、マルチチャネルコンパイルから出力されるメタビットの数は、区別されるべきオブジェクトのタイプの数及び実行されるべきオブジェクトの各タイプのレベルの数に依存して、1、2、4、又は8であってよい。

【0046】以下に述べられるように、各コマンド命令は、単一のスキャンラインにのみ関連し、そのスキャンラインの可変長を定義する。従って、マルチチャネルコンパイルの出力をIOTのページ及びラインへ適切に同期することが出来ないで、並行入力ポート1650がページ同期(sync)信号、ライン同期信号及びIOTクロック信号を受信してマルチチャネルコンパイル1638へ入力する。

【0047】出力FIFO1640及び1642は、並行出力ポートドライバ1652及び1654へ夫々接続される。これらの並行ポート1652及び1654からの出力がIOT1700の入力ポートへ接続される。

【0048】データFIFO1642が、プリント用に処理されるべきデータとしてIOT1700により受信されることと理解されるべきである。しかしながら、本発明は、これと理解されるべきである。図3に示される前に、データFIFO1642からの各バイトのプリントデータが出力メタビットFIFO1640からの1、2、4、又は8メタビットにより同期して伴われるということを要する。これらのメタビットは、プリントされる前に各データバイトへなされるべき処理を指定する。説明されたように、この最速処理は、各データバイトが抽出されたオブジェクトに依存して、各データバイトと共に実行する。図34に示されるように、IOTコンローラ160の好適な実施の形態において、IOT1700は、IOT1700へ送信されるメタビットデータにより選択的に制御されるオブジェクト最適化されたサブシステムの大きさを含む画像処理システム1710を有する。このよう

(8) 神岡2000-153640

14

【0056】種々の画像処理サブシステム及びプリントマルチプレクサ1712-1730が交互にオブジェクト最適化プリントシステム全体に亘って、多くの他のポートで表れてもよいことが理解されるべきである。例えば、図示されるように、オブジェクト最適化画像処理サブシステム及びプリントマルチプレクサ1712-1730をオブジェクトタイプが決定された後は何時でも、PFA分解手段130内、IOTコマンド命令発生手段140内のようなオブジェクト最適化ESS100に、IOTコンローラ160の一部として、或いはIOT1700内に配置することができ、また、これらのサブシステムをシステム全体に分散配置できる。

【0057】更に、これらのオブジェクト最適化画像処理サブシステム及びプリントマルチプレクサ1712-1730は、ハードウェア又はソフトウェア或いはこれら二つの組み合わせで実行される。あらゆる場合において、サブシステムは、異なる手順及びそれらが処理しているオブジェクトのタイプに基づくデータ型を選択している。従って、異なるオブジェクトタイプに対して異なる処理結果を生成する。

【0058】また、カラー測定デバイス190は、測定較正プリントと連結されるメタビット情報を使用して、オブジェクト最適化ベースで画像処理サブシステム1712-1722を自動的に調整し安定化することができ、例えば、図35に示されるように、較正プリント測定する出力カラー測定デバイス190は、較正プリントの実際の状態を指示する信号を出力する。メタビット情報及び出力カラー測定デバイスからの出力信号に基づいて、オブジェクト最適化出力校正サブシステム1732は、オブジェクト最適化プリント再生曲線サブシステム1716における全体のセットから一つのトーン再生曲線(TRC)ルックアップテーブルを調整する。画像処理システム調整(機能)に対して行われる訂正は、オブジェクト最適化出力校正サブシステム1732がメタビットチャネルでエンコードされたタグにより制御されるので、異なるオブジェクトに対して異なる。

【0059】図3は、マルチチャネル1630の一つの例示の実施の形態を示す。図3に示されるように、メタビットFIFO1626は、メタビットアンパッカー1631に接続され、8ビット幅メタビットを少なくとも二つの4ビット幅メタビットニブルに変換する。メタビットアンパッカー1631からの出力は、四つのカラー/メタビットレジスタ1634-1637の各々に接続される。メタビットアンパッカー1631からの出力がIOT制御及びコマンド/マスクチャネルプロセッサ1680により出力されるロードストレーン(LD)信号に依り、カラー/メタビットレジスタ1634乃至1637のそれぞれにロードされる。同様に、一定のカラーチャネルFIFO1624からのデータ入力、4個のカラー/メタビットレジスタ1634乃至1637へ提供され、ロード

13

オブジェクト最適化レーザパワー選択サブシステム1724を制御してそのバイトの画像データを形成するため低いパワーセットポイントを使用する。

【0052】これらのセットポイントの各々に関連するレーザパワーが他のセットポイントの関連において設定されてもよいことが理解されるべきである。例えば、低いセットポイントは、あらゆる他のセットポイントから独立して設定される。次に、高いセットポイントは、低いセットポイントよりも10%高く設定される。或いは、セットポイントが互いから独立して設定されてもよい。

【0053】また、多くのプリントが”書き込み”タイプであり、それらはデキスト/ラインアートのような現象されたマークであってもよいあるメディアの領域を露光することを理解すべきである。光がより高くなると、露光によりマークがより太くなる。他のプリントは、マークを囲む領域を露光する”白書き込み”タイプであり、これらの露光領域が現像されず、未露光領域が現像される。白書き込みシステムにおいてより太いマークを作成することは、デキスト/ラインアートを囲む白領域により低い露光が必要である。

【0054】更に、プリントシステムが入力スキャンのような他の同期データソースを有する場合、メタビットチャネルは、プリントに先立って、これらのデータストリームをデータマルチプレクサ1730を介してIOTコンローラ160から入ってくるプリントデータとミックスするために使用され得る。匹敵する装置がIOTコンローラ160においてミキシングを実行するために使用され得る。

【0055】上述の本発明による装置の例示の実施の形態において、メタビットは、出力が予めロードされている1セットのメタビットマップレジスタ1740間で選択することにより、オブジェクト毎ベースで前記複数の画像処理サブシステム1712-1738を制御して、オブジェクト最適化画像処理モジュール及びプリントマルチプレクサ1712-1730の各々で指定されたサブセットの使用可能ラインを選択する。例えば、4メタビットが実行される場合、それらは、1セットの16個のレジスタから選択して使用され得る。一方、これらのレジスタは、各々十分に制御でき、画像処理サブシステム及びプリントマルチプレクサ1712-1730のサブセットから選択されることが必要なあらゆるサイズのものであり得る。このように、各メタビット値の意味は、完全にプログラム可能であり、それが選択するレジスタの内容を変化することにより簡単に変更され得る。一方、そのレジスタは、そのメタビットによりタグ付けされるオブジェクトの特定のタイプに対して最も可塑性の高いレジスタを実行するために、画像処理サブシステム及びプリントマルチプレクサ1712-1730の完全なセットから選択してもよい。

12

に、メタビットチャネルは、各バイトのプリントデータに入ってくるオブジェクトに依存して異なるように処理させる。

【0049】例えば、メタビットは、多数のハードトーン発生器1712間で選択できる。一つのハードトーン発生器は、高空間周波数ハードトーン発生器を使用すべきカラーのテクニクのようなオブジェクトからくると、タグ付けされたデータバイトと共に使用され得る。他の物は、写真や絵画等から得られるデータバイトと共に使用される低空間周波数カラー画像ハードトーン発生器であってよい。各オブジェクト毎のハードトーンストリーンの角度の制御のようなハードトーン発生に対するより微細な制御は、メタビット選択可能である。ハードトーンの選択に加えて、種々のオブジェクトのそれらのエッジで互いに対話する事に起因する問題を処理する、オブジェクト最適化カラー変換サブシステム1714、オブジェクト最適化トーン再生曲線変換サブシステム1716、オブジェクト最適化空間フィルタリングサブシステム718及びオブジェクト最適化トラッピングサブシステム1720、及び異なるオブジェクトにおいて異なるように表れる種々のプリントシステムの問題を制御しようとするオブジェクト最適化エンジン/ノイズ抑制サブシステム、並びに他のオブジェクトに依存する可変結果を生成するテンムのような画像処理システム1710においてメタビットによって多くの追加の画像処理サブシステムが制御され得る。

【0050】IOT1700は、画像処理システム1710内にレーザパワー選択サブシステム1710のレーザ選択サブシステム1724を制御して、画像データの各特定のバイトに関連するメタビットに依存して、異なるセットポイントの間で選択する。特に、オブジェクト最適化レーザパワー選択サブシステム1724は、画像データのバイトがハードトーンオブジェクト又はデキスト又はラインアートオブジェクト内に在るかに基づいて、そのバイトの画像データをレンダリングするために使用されるレーザパワーのセットポイントを変更する。オブジェクト最適化レーザパワー選択サブシステム1724は、オブジェクトをレンダリングするために最も適切なレーザパワーを提供するように複数のセットポイント間での選択を可能とする。

【0051】例えば、メタビット値は、そのバイトの画像データがデキスト又はラインアートオブジェクトの部分であることを指示する場合、メタビット値がオブジェクト最適化レーザパワー選択サブシステム1724を制御してそのバイトの画像データを形成するために割り当てられべきセットポイントを使用する。或いは、メタビット値は、そのバイトの画像データがハードトーンオブジェクト内にあることを指示する場合、メタビット値が

30

2は、オブジェクト最適化出力校正サブシステム1732は、オブジェクト最適化プリント再生曲線サブシステム1716における全体のセットから一つのトーン再生曲線(TRC)ルックアップテーブルを調整する。画像処理システム調整(機能)に対して行われる訂正は、オブジェクト最適化出力校正サブシステム1732がメタビットチャネルでエンコードされたタグにより制御されるので、異なるオブジェクトに対して異なる。

40

1に接続され、8ビット幅メタビットを少なくとも二つの4ビット幅メタビットニブルに変換する。メタビットアンパッカー1631からの出力は、四つのカラー/メタビットレジスタ1634-1637の各々に接続される。メタビットアンパッカー1631からの出力がIOT制御及びコマンド/マスクチャネルプロセッサ1680により出力されるロードストレーン(LD)信号に依り、カラー/メタビットレジスタ1634乃至1637のそれぞれにロードされる。同様に、一定のカラーチャネルFIFO1624からのデータ入力、4個のカラー/メタビットレジスタ1634乃至1637へ提供され、ロード

ビットアンパッカーにより使用される2つ(必要に応じて又は3つ又は4)メタビットアンパッキング方式を定義する。即ち、メタビットデータのビットがどのようにメタビットパケットへ分解されるかを決定するための追加の方法が必要な場合、第1および第2のビットが共にメタビットアンパッキングの4つまでのタイプを提供するために使用される。

【0066】第1の構成レジスタ1681のビット3は、唯一つの1ビット幅メタビットチャネルが10Tへ提供される場合に、メタビットFIFOをデイスイネーブル(使用不能)にするために使用される。第1の構成レジスタ1681のビット4及びビット5は、診断モードで使用されてデータをFIFO及び内部レジスタから読み出す。第1の構成レジスタ1681のビット6は、マルチチャネルコンパインがノーマルモードか診断モードかを指示するために使用される。第1の構成レジスタ1681のビット7は、白(例えば、紙の自然なバックグラウンドカラー)が提供されるのが全て0から成るデータパ* SEL アクセスされるレジスタ

【表2】

SEL	アクセスされるレジスタ
0	ビ*データFIFOへの137/142、141/142、140
1	ビ*データFIFOへの137/142、141/142、141
2	ビ*データFIFOへの137/142、141/142、142
3	ビ*データFIFOへの137/142、141/142、143
4	ビ*データFIFOへの137、142、143
5	ビ*データFIFOへの137、142、143、144
6	ビ*データFIFOへの137、142、143、144、145
7	ビ*データFIFOへの137、142、143、144、145、146
8	ビ*データFIFOへの137、142、143、144、145、146、147
9	ビ*データFIFOへの137、142、143、144、145、146、147、148
10	ビ*データFIFOへの137、142、143、144、145、146、147、148、149
11	ビ*データFIFOへの137、142、143、144、145、146、147、148、149、150
12	ビ*データFIFOへの137、142、143、144、145、146、147、148、149、150、151
13	ビ*データFIFOへの137、142、143、144、145、146、147、148、149、150、151、152
14	ビ*データFIFOへの137、142、143、144、145、146、147、148、149、150、151、152、153
15	将来使用のために147、148、149、150、151、152、153

診断SEL値に対するレジスタアクセスデコード表

【0068】構成レジスタ1681のビット6がハイ(即ち、ノーマルモード)の時、マルチチャネルコンパイン1630は、ノーマルプリントデータを発生し、それをデータFIFO1642へ送信される。

【0069】表3は、第2の構成レジスタ1684のビット割り当てを示す。ビット7及びビット8は、10Tイ ※

ビット	ハイ	ロー
8	P同相立ち上がり	P同相立ち上がり
7	L同相立ち上がり	L同相立ち上がり
6	偽P同相立ち上がり	偽P同相立ち上がり
5	偽L同相立ち上がり	偽L同相立ち上がり
4	将来使用のためリザーブ	将来使用のためリザーブ
3	将来使用のためリザーブ	将来使用のためリザーブ
2	将来使用のためリザーブ	将来使用のためリザーブ
1	将来使用のためリザーブ	将来使用のためリザーブ

構成/ステータスレジスタ1(CSR1)ビット割り当て

【0070】再び図4を参照すると、コマンド/マスクコントロール1686は、16ビットコマンド命令及び1ビットマスクチャネル入力を受信する。表4及び5は、ノーマルノリビートモードビットがノーマルにセッ トされる(即ち、ビット15が0へセットされる)と、16ビットコマンド命令に対するビット割り当てを示す。

【表4】

ネーブル信号に基づいてカラー/メタビットレジスタの 一つにロードされる。

【0060】サンプリングチャネルFIFO1620からの18 ビット幅入力データは、サンプリングチャネルアンパッカー 1632Aに入力され、2バイト幅サンプリングチャネル データ及び2ビット"有効-バイト"データを2個の単一バ イト出力に変換され、それらは、連続的に内部サンプリング FIFO1632Bへ出力されるか、或いはそれらの対応する"有効-バイト"はこのバイトが無効であるということ を指示する場合は、破棄される。

【0061】コマンドFIFO1628からの2バイト 幅出力及びマスクFIFO1622からの1ビット幅出力 は、コマンド/マスクチャネルプロセッサ1680へ直 接入力される。また、コマンド/マスクチャネルプロセ ッサ1680へは、並行入力ポート1650を介して10 Tから受信されるページ同期、ライン同期及び10Tクロ ックが入力される。コマンド/マスクチャネルプロセッサ 1680制御信号をカラー/メタビットマルチプレクサ 1633及び出力マルチプレクサ1639、及びFIFO1 620-1628への4つの出力信号を出力する。4 つの出力信号は、カラー/メタビット読み出し(RD) 信号、サンプリング読み出し信号、マスク読み出し信号及びコマ ンド命令読み出し信号よりなり。対応するFIFO1620- 1628がコマンド/マスクチャネルプロセッサ168 0からこれらの信号の一つを受信するとすぐ、対応する FIFO(単数又は複数)は、対応するチャネル上の次のデ ータを読み出す。FIFO(単数又は複数)1620乃至1 628は、バスインターフェース1610によるドライ (dry)ランニングを防止し、計算された優先順位に従 ってメモリ150に格納されたチャネルデータでそれらを充填す る。

【0062】コマンドチャネルFIFO1628から受信さ れたコマンド命令に基づいて、コマンド/マスクチャネ ルプロセッサ1680は、カラー/メタビットレジスタ 1634乃至1637の対応する一つに対してロードイ ネーブル信号LEを発生する。コマンド/マスクチャネ ルプロセッサ1680の一つがイネーブルされると、同 時にそれは、一定カラーFIFO1624からの9ビット幅 一定カラーデータ及びメタビットアンパッカー1631*40

*からの4ビットまでのメタビットデータをロードする。 或いは、カラーデータは、別個のコマンドでメタビット から独立してロードされてよい。しかしながら、カラ ー/メタビットレジスタ1634乃至1637は、それ らが絶えず同時にロードされるので、何時も共に参照さ れる個別のレジスタであることが好ましい。

【0063】また、コマンド/マスクチャネルプロセッ サ1680は、出力マルチプレクサ1639及びメタビ ットレジスタ1638へ入力されるべきカラー/メタビ ットレジスタ1634乃至1637の一つを選択するた めに、カラー/メタビットマルチプレクサ1633に対 して制御信号を発生する。更に、コマンド/マスクチャ ネルプロセッサ1680は、カラー/メタビットマルチ プレクサ1633からの出力と内部サンプリングチャネルFIFO 1632Bからの出力との間で選択するために出力マル チプレクサ1639に対して制御信号を発生する。出力 マルチプレクサ1639は、カラー/メタビットマルチ プレクサ1633により出力される8ビット幅カラーデ ータ及び第1番目のビットのメタデータを出力する。こ れは、データFIFO1642へ入力される。同時に、メタ ビットレジスタ1638は、第2番目-第4番目のメタ ビットを格納し、それらを出カメタビットFIFO1640 へ出力する。好ましい実施の形態においては、4ビ ット幅メタビットコンプリの全てを含むことは必要ない事 が理解されるべきである。或いは、10Tへ提供されるメ タビットの数は、1、2、又は4、或いは8であっても よい。このように、図2及び3に示される例示の実施の 形態は、10Tへ提供されるメタビットの数を4に制限す るものとして解釈されるべきではない。

【0064】図4は、コマンド/マスクチャネルプロセ ッサ1680の内部機能ブロックを示す。図4におい て、コマンドチャネルライン及びマスクデータチャネル ラインを除いて、制御ラインのみが示されている。コマ ンド/マスクチャネルプロセッサ1680において、ニ 一つの構成レジスタ1681及び1684、レジスタCSR 0及びCSR1、が表4に示される特徴(機能)の制御を 提供する。

【表1】

ビット	ハイ	ロー
8	将来の使用のためにリザーブ	将来の使用のためにリザーブ
7	白=00	白=00
6	ノーマルモード	ノーマルモード
5	レジスタ読み出し	レジスタ読み出し
4	FIFO1-1に対して147、148をわ	FIFO1-1に対して147、148をわ
3	メタビットFIFOを使用	メタビットFIFOを使用
2	メタビットアンパックをオン	メタビットアンパックをオン
1	メタビットアンパック1をオフ	メタビットアンパック1をオフ

構成/ステータスレジスタ0(CSR0)ビット割り当て

コマンドビット	フィールド記述子
0	RepCnt0
1	RepCnt1
2	RepCnt2
3	RepCnt3
4	RepCnt4
5	RepCnt5
6	SCRes0
7	SCRes1
8	SrcB0
9	SrcB1
10	SrcA0
11	SrcA1
12	Cur0
13	Cur1
14	LeColor
15	Repeat/Normal

ノーマルモード (ビット15=0)

【表5】

LaColor	コマンド
0	SrcBをロードしない
1	SrcBをローディング時に、SrcAが17-19ビットをFIFOからロードする。この場合、SrcAが17-19ビットをFIFOからロードする。
2	SrcAが17-19ビットをFIFOからロードする。
3	SrcAが17-19ビットをFIFOからロードする。
4	SrcAが17-19ビットをFIFOからロードする。
5	SrcAが17-19ビットをFIFOからロードする。
6	SrcAが17-19ビットをFIFOからロードする。
7	SrcAが17-19ビットをFIFOからロードする。
8	SrcAが17-19ビットをFIFOからロードする。
9	SrcAが17-19ビットをFIFOからロードする。
10	SrcAが17-19ビットをFIFOからロードする。
11	SrcAが17-19ビットをFIFOからロードする。
12	SrcAが17-19ビットをFIFOからロードする。
13	SrcAが17-19ビットをFIFOからロードする。
14	SrcAが17-19ビットをFIFOからロードする。
15	SrcAが17-19ビットをFIFOからロードする。

ノーマルコマンドデコード表

【0071】ノーマルモードにおいて、表4に示される 50 ように、ビット0-5は、6ビットカウント値を表す。

カウント値は、このコマンドが連続するピクセルの幾つ有効かを指示する。表5は、ビット6及び7に対する値に依存して、サンプリングクロックが1、2、3又は4で割られる。サンプリングクロックを割ることによって、出力画像の解像度が内部サンプリングFIFO1632Bの現在のカラーを繰り返し出力させることにより変化され得る。即ち、サンプリングクロックを4で割れば、コマンドが選択されるようなビット6及び7の値が共に0の時、内部サンプリングFIFO1632Bが各クロックサイクル毎に4回繰り返し出力される。従って、例えば、400スポット/インチ (spi) 解像度のプリンタは、何ら追加の処理が必要無く、サンプリングマップを400spi、200spi、133spi、及び100spiでプリントすることができる。

【0072】ビット8及び9は、カラーレジスタ選択値を表し、またビット10及び11は、カラーレジスタ選択値を表す。表5に示されるように、カラーレジスタ選択値及びカラーレジスタ選択値は、カラー/メタビットレジスタ1634-1637 (又は内部サンプリングFIFO1632B) の何れがデータFIFO1642へコンパイルナによってバスされるかを指示する。カラーレジスタ選択値がカラー/メタビットレジスタ1634、1635及び1636又は内部サンプリングFIFOの間で選択し、且つカラーレジスタ選択値がカラー/メタビットレジスタ1634-1637間で選択する。二つの選択値A及びBがあり、それにより1ビットチャネルがそれらの間の切り替えるために使用される。このモード、ビット12及び13により指定され、且つなんらの追加の命令オーバーヘッド無しにA及びBにより指定される二つのレジスタの間で迅速に切り替えるために使用される。

【0073】ビット12及び13は2ビットコンパイルナ制御値を表す。コンパイルナ制御値は、6ビットカウント値によって指示される次の値のピクセルに対してマルチチャネルコンパイルナ1630によってデータFIFO1642へ出力されるべきカラーデータを指示する。表5に示されるように、コンパイルナ制御値は、出力されるべきカラーがカラーレジスタ選択値、又は、カラーレジスタ選択値によって選択されるレジスタ、又はマスクチャネルによって更に制御されるその二つのレジスタの組み合わせから来るのかを指示する。

【0074】最後に、表5に示されるように、ビット14は、カラーチャネルFIFO1624から出力される次のカラーデータをカラーレジスタ選択値によってカラー/メタビットレジスタ1634-1637の一つにロードされるべきであるかを指示する。

【0075】動作において、新たなノーマルモードコマンドがコマンドチャネルFIFO1628から入力されると、ロードカラービット14は、カラーチャネルFIFO1624に格納された次の8ビットカラーデータバイトがカラーレジスタ選択ビット8及び9によって指示されるカラー/メタビットレジスタ1634-1637にロ

ードされるべきか否かを決定するためにデコードされる。カラーレジスタ選択ビット8及び9がカラー/メタビットレジスタ1634-1636 (カラー/メタビットレジスタ0-2) を指示する時、カラーは、コマンドの第1のクロックサイクルでのみカラーチャネルFIFO1624からロードされる。

【0076】しかしながら、カラーレジスタ選択ビット8及び9がカラー/メタビットレジスタ1637 (カラー/メタビットレジスタ3) を指示すると、新たな一定カラーデータがこのコマンドの各クロックサイクル毎に、カラーチャネルFIFO1624からカラー/メタビットレジスタ1637にロードされる。このように、実際に、一定カラーチャネルが低速サンプリングチャネルとして使用され得る。幾つかの複雑性の高い画像において、スキャンラインにおける一定カラーデータは、迅速に変化するもので (即ち、1-8画素の全てを変化する) ので、新たなカラー毎に新たなコマンドワードを発生し、格納し、伝送し、そしてデコードするために必要な処理が通剰になり、その結果、バイトマップ表現に相対して負の圧縮となる。

【0077】この効果的な負の圧縮は、一定カラーのストリーミングを一定カラーチャネルFIFO1624へ提供し、カラーFIFO1624から一定カラーのストリーミングの一つの一定カラーバイトをクロックサイクル毎に搬出し、それをカラー/メタビットレジスタ1637へロードすることにより回避され得る。サンプリングされたデータチャネルFIFO1620におけるように、一定カラーチャネルFIFO1624が一回に4バイトではなく、1回に2バイトをロードできるように過ぎない。このように一定カラーチャネルFIFO1624を使用することは、一定カラーチャネルのストリーミングがサンプリングデータチャネルのストリーミングより低くなる。しかしながら、この「スローサンプリング」方法は、レギュラーサンプリングデータチャネル (以下に詳述される) に使用されるDMAポインタを発生する時に必要なオーバーヘッドを避ける。このように、この「スローサンプリング」チャネルは、迅速に変化する一定カラーのストリーミングに対しては最も有用であるが、大きな画像に対しては使用されない。

【0078】次に、コンパイル制御ビット12及び13は、カラーデータソースとしてカラー/メタビットレジスタ1634-1637 (又は内部サンプリングFIFO1632B) の何れが使用されるかを決定するためにカラー選択レジスタA及びBの何れが使用されるかを決定すると共に、マスクチャネルからのマスクデータがマルチチャネルコンパイルナ1630からデータFIFO1642へのデータフローを更に制御するために使用されるか否かを決定するためにデコードされる。勿論、ロードカラービット14及びカラー選択レジスタビット8と9が「スローサンプリング」カラーチャネルとしてカラー/メタビットレジスタ1637を使用することを指示すると、制

【0079】この効果的な負の圧縮は、一定カラーのストリーミングを一定カラーチャネルFIFO1624へ提供し、カラーFIFO1624から一定カラーのストリーミングの一つの一定カラーバイトをクロックサイクル毎に搬出し、それをカラー/メタビットレジスタ1637へロードすることにより回避され得る。サンプリングされたデータチャネルFIFO1620におけるように、一定カラーチャネルFIFO1624が一回に4バイトではなく、1回に2バイトをロードできるように過ぎない。このように一定カラーチャネルFIFO1624を使用することは、一定カラーチャネルのストリーミングがサンプリングデータチャネルのストリーミングより低くなる。しかしながら、この「スローサンプリング」方法は、レギュラーサンプリングデータチャネル (以下に詳述される) に使用されるDMAポインタを発生する時に必要なオーバーヘッドを避ける。このように、この「スローサンプリング」チャネルは、迅速に変化する一定カラーのストリーミングに対しては最も有用であるが、大きな画像に対しては使用されない。

【0078】次に、コンパイル制御ビット12及び13は、カラーデータソースとしてカラー/メタビットレジスタ1634-1637 (又は内部サンプリングFIFO1632B) の何れが使用されるかを決定するためにカラー選択レジスタA及びBの何れが使用されるかを決定すると共に、マスクチャネルからのマスクデータがマルチチャネルコンパイルナ1630からデータFIFO1642へのデータフローを更に制御するために使用されるか否かを決定するためにデコードされる。勿論、ロードカラービット14及びカラー選択レジスタビット8と9が「スローサンプリング」カラーチャネルとしてカラー/メタビットレジスタ1637を使用することを指示すると、制

【0079】この効果的な負の圧縮は、一定カラーのストリーミングを一定カラーチャネルFIFO1624へ提供し、カラーFIFO1624から一定カラーのストリーミングの一つの一定カラーバイトをクロックサイクル毎に搬出し、それをカラー/メタビットレジスタ1637へロードすることにより回避され得る。サンプリングされたデータチャネルFIFO1620におけるように、一定カラーチャネルFIFO1624が一回に4バイトではなく、1回に2バイトをロードできるように過ぎない。このように一定カラーチャネルFIFO1624を使用することは、一定カラーチャネルのストリーミングがサンプリングデータチャネルのストリーミングより低くなる。しかしながら、この「スローサンプリング」方法は、レギュラーサンプリングデータチャネル (以下に詳述される) に使用されるDMAポインタを発生する時に必要なオーバーヘッドを避ける。このように、この「スローサンプリング」チャネルは、迅速に変化する一定カラーのストリーミングに対しては最も有用であるが、大きな画像に対しては使用されない。

【0078】次に、コンパイル制御ビット12及び13は、カラーデータソースとしてカラー/メタビットレジスタ1634-1637 (又は内部サンプリングFIFO1632B) の何れが使用されるかを決定するためにカラー選択レジスタA及びBの何れが使用されるかを決定すると共に、マスクチャネルからのマスクデータがマルチチャネルコンパイルナ1630からデータFIFO1642へのデータフローを更に制御するために使用されるか否かを決定するためにデコードされる。勿論、ロードカラービット14及びカラー選択レジスタビット8と9が「スローサンプリング」カラーチャネルとしてカラー/メタビットレジスタ1637を使用することを指示すると、制

【0079】この効果的な負の圧縮は、一定カラーのストリーミングを一定カラーチャネルFIFO1624へ提供し、カラーFIFO1624から一定カラーのストリーミングの一つの一定カラーバイトをクロックサイクル毎に搬出し、それをカラー/メタビットレジスタ1637へロードすることにより回避され得る。サンプリングされたデータチャネルFIFO1620におけるように、一定カラーチャネルFIFO1624が一回に4バイトではなく、1回に2バイトをロードできるように過ぎない。このように一定カラーチャネルFIFO1624を使用することは、一定カラーチャネルのストリーミングがサンプリングデータチャネルのストリーミングより低くなる。しかしながら、この「スローサンプリング」方法は、レギュラーサンプリングデータチャネル (以下に詳述される) に使用されるDMAポインタを発生する時に必要なオーバーヘッドを避ける。このように、この「スローサンプリング」チャネルは、迅速に変化する一定カラーのストリーミングに対しては最も有用であるが、大きな画像に対しては使用されない。

【0078】次に、コンパイル制御ビット12及び13は、カラーデータソースとしてカラー/メタビットレジスタ1634-1637 (又は内部サンプリングFIFO1632B) の何れが使用されるかを決定するためにカラー選択レジスタA及びBの何れが使用されるかを決定すると共に、マスクチャネルからのマスクデータがマルチチャネルコンパイルナ1630からデータFIFO1642へのデータフローを更に制御するために使用されるか否かを決定するためにデコードされる。勿論、ロードカラービット14及びカラー選択レジスタビット8と9が「スローサンプリング」カラーチャネルとしてカラー/メタビットレジスタ1637を使用することを指示すると、制

た、各チャネルに対するデータが開始するメモリ150にアドレスを格納するレジスタ1614の一セットのレジスタがある。メモリサイズをセーブするために、マスクチャネル及びサンプリングチャネルは、データ取り出しにおいて、一レベルの間接処理 (indirection) を使用する。単一のアドレスポインタレジスタを使用する代わりに、マスク及びサンプリングチャネルの各々は、3個のレジスタを有する。第1のものは、表 (テーブル) が格納されているメモリ150中のアドレスをポインタする。この表は、アドレス/サイズ対のリストを含み、これらの対の各々は、次のサンプリング又はビットマップセクションが見つけれられるメモリ150の部分に指定する。サンプリングチャネル制御1612は、次のアドレスをレジスタ1614のサンプリングアドレスレジスタへ取出し、且つサンプリングデータのそのブロックのそのサイズをレジスタ1614のサンプリングサイズレジスタへ取り出す。次に、バスインターフェース1610は、正確な量のデータが取り出されるまで、メモリ150のその部分から取り出されることができ、その後、それは、サンプリングデータ中の次のアドレス/サイズ対を取り出す。マスクチャネルは、同様に処理される。

【0091】取出しは、絶対ワード境界で発生するので、実際の第1の有効バイトのデータがワード内に発生するサンプリングデータの場合に可能である。このように、サンプリングチャネル制御1612は、取り出されようとしたメモリ150の中のバイトアドレスを実際に取り出されたメモリ150中のワードアドレスと比較して、“無効”タグで取り出された第1のワードの3バイトまでタグ付けできなければならない。サンプリングチャネルアンパッカー1632Mは、後で、これらの無効バイトを内部サンプリングFIFO1632Bへロードするのでなくで破棄できる。

【0092】バスインターフェース1610の他の対は、データレートを等しくするためのパッドアームメモリ1616、ローレベルバス (Shub) カード機能処理するカードコントローラ1618、及びSバスリクエスタアドレスマップピング及びデコーディングを処理するデコード1615を含む。デコード1615の一つの重要な機能は、状態情報を送信受信できるIOT170のシリアルポートへのインターフェースを提供することである。

【0093】図6は、IOTコントローラ260の第2の例示の実施の形態を示す。図6に示されるように、バスインターフェース1610と本質的な動作及び構成が同様である。バスインターフェース2610は、バス114から4バイトバスを受信し、32ビットデータストリームをコンプレッサ (圧縮器) /デコンプレッサ (圧縮解除器) コントローラ2650及び統合コンパイン/チャネルFIFO2630へ出力する。統合コンパイン/チャネルFIFO2630は、マルチチャネルコンパイン

【0087】図5に示されるように、バスインターフェース1610は、バスFIFO1611、サンプリングチャネル制御1612、スローチャネル制御1613、レジスタ1614、デコード1615、パッドアームメモリ1616、チャネル制御ロジック1617及びカードコントローラ1618を有する。

【0088】バスインターフェース1610の基本動作は、直接メモリアクセス (DMA) を使用するバス上のバックケットを検索して、それらを5個のFIFOチャネル1620-1628の一つへ書込むことである。バスインターフェース1610は、各ペーজの前に1度プログラムされ、次にFIFOチャネル1620-1628を充填するために独立して動作する。チャネル制御ロジック1617は、FIFOチャネル1620-1628の各々が充填される優先順位を決定するために使用される。これらは、殆ど空である時に発生されるFIFO1620-1628の各々から、各チャネルに対するプログラム可能“重み”及び信号のチャネルの内の三つに対するインターリーブオペレーションに基づく。重みは、チャネルが現在最も高い優先順位として残る行の回数を示す。

バスインターフェース1610の第1の実施の形態の操作方式は、優先順位の三つのサブセットに基づく。優先順位サブセット1は、常に最も高い優先順位であり、後で経過されるように、新たなサンプリング及びマスクチャネルポインタの取出しを含む。優先順位サブセット2は、優先順位が交替する。優先順位サブセット3は、ラウンドロビン取出しコマンドであり、それは、マスク、カラー、メタビット及びコマンドチャネルデータを取り出す。

【0089】この操作方式の背景理由は、以下の通りである。ポインタ (サブセット1) は、小さく、多くないが、以下に述べられるように、重要な取出しである。サンプリングチャネルデータ (サブセット2) は、システムを介して最も大きなバンド幅を要求する可能性のある生バイトである。他のチャネルは、“スローチャネル”と呼ばれる。その理由は、それらのチャネルFIFO1622-1628が32ビット幅よりも少ない幅を有するからである。それらを取り出すと、取り出されたワードをバスFIFO1611へ書込むことが必要であり、それに

より、それらは、スローチャネル制御1613によって並行にそれらの個別のFIFOチャネル1622-1628上にアンパッキングされることができる。この動作の間に、他のバス取り出しがサンプリングチャネルFIFO1620に対して並行に発生する可能性がある。

【0090】各チャネルに対して取出しのバス転送サイズを制御する事ができる。レジスタ1614の一つは、各チャネルに対するバス転送サイズを格納する。ま

1630をデータチャネルFIFO1620-1628と統合する。コンプレッサ/デコンプレッサコントロール2650は、データをメモリ150からIOTコントローラ260へ転送する時に圧縮されたデータの使用を可能とすると共に、メモリ150から転送されるメタビット情報によって制御される。コンプレッサ/デコンプレッサコントロール2650は、このように、メタビット情報にどのタイプの圧縮がデータの受信されたブロックへ適用されたかを指示することを可能にする。

【0094】従って、オペラティブタイプ (即ち、データが、“JPG” (ジョイントフォトグラフィックエクスパート) データを使用して最適に圧縮されたカラー画像データか、白/黒ビットマップデータ及びラン長エンコーディング又はCCITTのような2値圧縮データ) を使用して最適に圧縮され得る他のデータタイプであるか) に基づいてデータ圧縮を最適化できる。従って、コンプレッサ/デコンプレッサコントロール2650へ入力されるデータのオペラティブタイプに基づき、データは、サンプリングコンプレッサ/デコンプレッサ2620又は2値サンプリングコンプレッサ/デコンプレッサ2640へ出力される。更に、スキヤナインターフェース116 (ディジタルカメラのような他のデバイスへのインターフェースであってもよい) は、コンプレッサ/デコンプレッサコントロール2650へ取り付けられることができ、それにより、リアルタイムで、スキヤナからのオンザフライデータが得られ、圧縮され、コンプレッサ/デコンプレッサコントロール2650へ転送され、圧縮解除され、統合コンパイン/チャネルFIFO2630の適切なチャネルへ入力される。このように、複写機や複写機/プリンタのスキヤナの場合、スキヤナ又は他の等価の物を介して画像の部分のオンザフライ挿入を可能とし、それによりオンザフライデータは、ページ画像のPDL記述を変える必要がなく、メタビットチャネルの制御下でページ画像へ結合され得る。更に、これにより、スキヤナ又は他の等価なデバイスを介して画像の部分のオンザフライ最適化を可能とする。例えば、画像の部分により高いレベルのレーザ光を使用してレンダリングされるべきであるならば、オペレータは、増加された露光レベルセットポイントを使用して、画像のどの部分がレンダリングされるべきかを識別できる。

【0095】サンプリング又は2値データが一旦圧縮解除されると、それは統合コンパイン/チャネルFIFO2630へ入力され、それは実質的に図2-4に関して上述されたように動作する。

【0096】更に、IOTコントロール260がカラー空間変換器 (トランスフォー) 2670を含むので、サンプリングカラーデータ及び一定カラーデータは、それが圧縮又は転送される前に、最適なカラー空間に配されることとができる。図6に示されるように、統合コンパイン/

【0097】図7に示されるIOTコントロールの第3の例示の実施の形態において、コンプレッサ/デコンプレッサ3650は、サンプリングコンプレッサ/デコンプレッサ3620を制御するためのみに使用される。図6の2値サンプリングコンプレッサ/デコンプレッサ2640は、より先進の階層で置き換えられた。この例示の実施の形態において、2値コンプレッサ/デコンプレッサ3640は、適応コンプレッサ/デコンプレッサ及び適応IOTコンプレッサ/デコンプレッサの何れかである。しかしながら、このIOTコントロール360は、一般的にIOTコントロールの第1及び第2の好適な実施の形態に関して上述されたように動作する。

【0098】最後に、図8は、IOTコントロール460の第4の例示の実施の形態を示す。この第4の例示の実施の形態において、統合コンパイン及びチャネルFIFO4630は、3カラー (24ビットワード) 又は4カラー (32ビットワード) を第1のカラー空間変換器4670及び第2のカラー空間変換器4675へ出力する。第1のカラー空間変換器4670及び第2のカラー空間変換器4675の各々は、IOT170へ出力される4カラー空間変換器の二つを発生するために使用される。一般に、カラー空間変換器の各々は、統合されたコンパイン及びチャネルFIFO4630、及びパッカー4672と4677の速度の2倍の速度で動作する。

【0099】例えば、第1のカラー空間変換器4670は、第1のクロックサイクルで、8ビットCカラー分層データバイトを出力し、第2のクロックサイクルで、8ビットMカラー分層データバイトを出力する。同様に、第1のクロックサイクルで、第2のカラー空間変換器4675は、8ビットYカラー分層データバイトを出力し、第2

チャネルFIFO2630は、24ビットワード (カラー当り3カラー×8ビット) をカラー空間変換器2670へ出力し、この変換器は、24ビット3カラーデータバイトのカラーデータへ変換し、次に、そのデータは、C、Y、M、Kカラー分層を要す4バイトとして、統合コンパイン/チャネルFIFO2630へ戻される。また、カラーデータは、バスインターフェース2610から直接的にカラー空間変換器2670へ入力される。どちらの場合も、メタビットデータは、カラー空間変換器2670を制御して現在のデータが関連する特定のオペラティブタイプに対する最適カラー空間変換を選択するために使用され得る。オペラティブ最適化カラー空間変換の部分として、最適トーン再生カーブ (TRC) がメタビットを使用して選択され得る。次に、図2-4に関連して上述されたように動作する、統合コンパイン/チャネルFIFO2630は、出力カラーデータ及びメタビットをIOT170へ出力する。IOT170の容量及びスピードに従って、最終データは、出力当り8ビットサイクルプラッス1-4メタビット又は出力当り32ビットサイクルプラッス1-4メタビットで出力され得る。

【0097】図7に示されるIOTコントロールの第3の例示の実施の形態において、コンプレッサ/デコンプレッサ3650は、サンプリングコンプレッサ/デコンプレッサ3620を制御するためのみに使用される。図6の2値サンプリングコンプレッサ/デコンプレッサ2640は、より先進の階層で置き換えられた。この例示の実施の形態において、2値コンプレッサ/デコンプレッサ3640は、適応コンプレッサ/デコンプレッサ及び適応IOTコンプレッサ/デコンプレッサの何れかである。しかしながら、このIOTコントロール360は、一般的にIOTコントロールの第1及び第2の好適な実施の形態に関して上述されたように動作する。

【0098】最後に、図8は、IOTコントロール460の第4の例示の実施の形態を示す。この第4の例示の実施の形態において、統合コンパイン及びチャネルFIFO4630は、3カラー (24ビットワード) 又は4カラー (32ビットワード) を第1のカラー空間変換器4670及び第2のカラー空間変換器4675へ出力する。第1のカラー空間変換器4670及び第2のカラー空間変換器4675の各々は、IOT170へ出力される4カラー空間変換器の二つを発生するために使用される。一般に、カラー空間変換器の各々は、統合されたコンパイン及びチャネルFIFO4630、及びパッカー4672と4677の速度の2倍の速度で動作する。

【0099】例えば、第1のカラー空間変換器4670は、第1のクロックサイクルで、8ビットCカラー分層データバイトを出力し、第2のクロックサイクルで、8ビットMカラー分層データバイトを出力する。同様に、第1のクロックサイクルで、第2のカラー空間変換器4675は、8ビットYカラー分層データバイトを出力し、第2

【0115】ステップS230において、利用可能リソースを出るページが到達すると、従来のPDL処理が現在のドキュメントの第1ページから開始される。状態及びグラフィカル演算子（オペレータ）パラメータの全てが0にリセットされる。第1ページからのグラフィカル演算子は、グラフィカル状態を正確に維持する限り、のみにおいて処理される。即ち、ビットマップ又はパイタマップは、書き込まれる。これは、不良ページに出会う前の最後の独立のページまで続く。その点から、グラフィカル演算子と画像データの両方が処理されるが、画像データは10へ出力されない。不良ページに到達する前に、処理された画像データは10へ出力される。続くページの処理とそれらの10への出力は、不良ページに出会った後の最初の独立ページまで続く。

【0116】この点から、オブジェクト最適化PDL分解手段130、及び107データ及びコマンド命令発生手段140は、現在のドキュメントの第1ページで再スタートし、全てのデータフィールド及びグラフィカル演算子を0にリセットする。オブジェクト最適化ESS100は、不良ページが発見された後の最初の独立ページまで画像データを発生することなく、オブジェクト最適化処理を続ける。この点から次の不良ページまで、PDL分解手段130及び107データ及びコマンド命令発生手段140は、上述のように動作する。次に、この処理は、全体のドキュメントがプリントされるまで、各不良ページ毎に繰り返される。

【0117】或いは、ステップS230において、二つの並行処理演算子が現在のドキュメントに対して初期化される。第1の処理演算子は、オブジェクト最適化処理であり得る。第2の演算子は、従来のバイトマップ/ビットマッププロセッサであり得る。オブジェクト最適化プロセッサは、第1の不良ページが発見されるまで続けることができる。この点で、第2のプロセッサは、不良ページが発見される前の最後独立ページまで画像データを発生することなく動作を開始する。この点から不良ページまで、画像データが発生されるが、107170へ出力されない。次に、不良ページ及び全ての続くページは、不良ページ後最初の独立ページまで、発生され、従来のプロセッサによって10へ出力される。不良ページ後の最初の独立ページに出会うと、オブジェクト最適化プロセッサは、不良ページから不良ページ後の最初の独立ページまでプリントデータを10へ出力することなく、グラフィカル演算子の分析を開始する。

【0118】この点から、オブジェクト最適化プロセッサは再びオブジェクト最適化されたデータを生成して107170へ出力し、次の欠陥ページに出会うまでそれを続ける。この時点で、従来のプロセッサのグラフィカル状態は、第2の欠陥ページの後最後の独立ページで第1の欠陥ページの後に第1の独立ページから更新される。次に、上述の処理は、現在のドキュメントの最後の

ページがプリントされるまで繰り返される。

【0119】しかしながら、ステップS220は、メモリリソースが十分であると決定すると、制御はステップS240へ続き、そこでスキャンライン及びコマンドがスキャンラインベースで発生される。次に、ステップS250において、リアルタイムチャネルデータが抽出されてメモリ150に格納される。

【0120】ステップS230及びS250は、ステップS260へ続き、そこで現在のドキュメントの更に処理の必要なページがあるか否かを決定する。もし在るならば、制御はステップS200へ戻る。しかしながら、更に処理するページがないならば、制御はステップS270へ続き、制御をステップS40へ戻る。

【0121】図11は、図9のステップS700のオブジェクト最適化分解及びレンダリングを使用して結合し及びプリントするための処理をより詳細に示す。図11に示されるように、ステップS700の結合しプリントする処理は、ステップS300で開始する。ステップS300において、ページの照会されたセット（即ち、現在のドキュメントのページ）の次の（第1）ページに対する圧縮されたデータが得られる。この次のページの圧縮されたデータは、メモリ150のRAM部分151からバス114を介してバスインタフェース1610によって得られる。

【0122】また、圧縮されたデータは、メモリ150の不揮発性メモリ部分152へ格納され得る。これは、クリエータがステップS300のオブジェクト指定圧縮及びレンダリングを使用してプリントデータを準備するための処理を繰り返す必要なしにドキュメントページのコピーを再プリントできることを望む場合に有用である。圧縮されたデータは、オブジェクトレンダリング機能を制御するためにメタビットデータを含み、それによりオブジェクト最適化能力が失われない。

【0123】次に、ステップS310において、現在のページの圧縮されたデータは、マルチチャネルコンバイナ1630を使用して圧縮解除され結合される。次に、ステップS330において、107コンローラ160へ提供されるメタビット情報は、107170によってプリントされるページを最適化するために、オブジェクト最適化レンダリング方法を選択し、同時入力ストリーム間で選択するように使用される。

【0124】即ち、マルチチャネルコンバイナ1630は、オブジェクトベースで、カラー空間変換、トーン再生カーブ、ハーフトーン発生器周波数及びスクリーン角、最大レーザパワーセットポイント及び/又は他の情報等のパラメータを決定するためにメタビット情報を使用する事ができる。最適データは、メタビットの制御下でデータへ適用される異なるタイプの処理に基づいて、決定されて107170へ出力される。

【0125】また、メタビット情報は、オブジ

ェクト最適化データを発生すると共に、他のメタビット情報がマルチチャネルコンバイナ1630によって107170へ出力されるようにマルチチャネルコンバイナ1630によって使用される。これらのメタビットは、且つ170の107170制御サブシステムへ出力され、且つこれらのメタビットは、プリントデータを更に最適化するために、カラー空間変換、トーン再生カーブ、最大レーザパワーセットポイント及び/又はハーフトーン発生器周波数及びスクリーン角を含み、

【0126】また、メタビット情報の全ては、107コンローラ160によってプリントデータと共に、107170のサブシステムへ転送される。更に、最大レーザパワーセットポイント、カラー変換、トーン再生等のオブジェクト最適化レンダリングの幾つかの情報は、マルチチャネルコンバイナ1630において或いは107170において、メタビットが他の機能を制御すると共に、ステップS300のオブジェクト指定レンダリング及び圧縮を使用してプリントデータを準備するための処理の間に適用され得る。

【0127】現在のページが、ステップS330において、107170によってオブジェクト最適化フォーマムでプリントされると、制御は、ステップS340へ進み、そこで現在のコピーの最後のページがプリントされたか否かを決定する。最後のページがまだプリントされていない場合、制御はステップS300へ戻る。しかしながら、現在のコピーの最後のページのプリントが完了すると、制御はステップS350へ続き、そこで希望のページでコピーの数が1だけインクリメントされ、次にステップS370へ進み、現在のページのポイントが現在のドキュメントの最初のページヘリセットされ、最後に、ステップS300へ戻る。しかしながら、ステップS350が最後のページがプリントされたと決定すると、制御はステップS380へ続き、制御をステップS90へ戻す。

【0128】図12は、図10のステップS210のオブジェクト最適化レンダリングタグを有するオブジェクトリストを構成するための処理をより詳細に示す。図12に示されるように、オブジェクトリストの構成は、ステップS400においてPDLドキュメントファイルを読み出すことによって開始する。次に、ステップS410において、次の（最初）の言語要素が現在の言語要素として得られ、解析される。現在の言語要素が解析された後、標準のグラフィックスインスタンフェースが呼び出される。標準のグラフィックスインスタンフェースに続くPDL及び他の動作の解析は、ひとつのPDLから他のPDLに変化する。これらの処理は公知である。

【0129】次に、ステップS420において、現在の言語要素は、それが現在のページの終わりを指示するか

発生手段140によって自動的に発生されることができ、或いは、オブジェクト最適化レンダリングタグを発生するための自動処理がデフォルトモードとして使用されると共に、ステップS20でドキュメントを準備する時に、ドキュメントクリエータは、PDLを使用してオブジェクトを指定する時のオブジェクト最適化レンダリングヒントを明示的に含むことが出来る。これらのクリエータ挿入レンダリングヒントは、オブジェクトタイプを明示的に定義すること、カラー空間変換又はトーン再生カーブを明示的に定義すること、ハーフトーンスクリーン周波数及び/又は角度を明示的にセットすること、そうでない場合は、PDL分解手段130及び/又は107データ及びコマンド命令発生手段140によって通常自動的に設定されるガムマトマップピン及び/又は他の望ましいオブジェクト最適化レンダリングパラメータに対する優先順位を指示する最大レーザパワー変換、即ちセットポイント、を明示的にセットすること、含む。

【0113】オブジェクト最適化レンダリングタグを含むオブジェクトリストは、ステップS210において現在のページに対して発生されているので、ステップS220において、利用できるモリのようなシステムリソースは、それらが破棄されなかったことを保証するためモニタされる。例えば、ページの直交する方向に対してある角度を持つて設定されたスライスを有する複雑なクリッピング領域のような、非常に複雑なページは、メモリ150のRAM部分151のメモリリソースが充分分である程度にオブジェクトリスト中に多くのオブジェクトが必要である。即ち、実際に、このページは、ナビゲート圧縮の問題を含む。

【0114】このように、各追加のオブジェクト毎に別のメモリを使用しないフォーマットバックモードが提供される。このフォーマットバックモードにおいて、現在のページは、サンプリングチャネルにレンダリングされ、且つシステムリソースにより、低下された解像度でレンダリングされてもよい。ステップS220が、リソースが十分でないことを決定すると、制御は、ステップS230へ進み、それは、フォーマットバックモードを使用して現在のページに対するプリントデータを準備する。一般に、フォーマットバックモードは、従来の107170に使用される従来のビットマップ/バイトマップを生成することに依存する。この場合、ページは、各オブジェクト毎のプリンタパラメータが最適化されるようにプリントされないが、少なくともそのページのプリントされ得る。更に、幾つかのPDLが、前のページ（即ち、ページが独立していない）を参照し、従来のビットマップ/バイトマップに対してディフォルトすることによってリソースのオーバーフローを処理することによって現在のページに対するプリントパラメータを定義するので、ステップS230におけるように、不良ページが前に定義されたパラメータに対してそのような参照を行うことが可能である。

否かを決定するために分析される。現在の音韻要素が現在のページの終わりを指示するならば、制御はステップS430へ進み、現在のページのクリッパー領域が検証される。クリッパー検証処理は、図19のステップS140-S1160に開連して更に詳細に述べられる。現在のクリッパー領域がステップS430において検証されると、制御はステップS440へ進み、制御がステップS220へ戻される。

【0130】現在の音韻要素がステップS420で現在のページの終わりを指示しないならば、制御はステップS450へ進む。ステップS450において、現在の音韻要素は、それがカラード演算子であるか否かを知るためにチェックされる。もしそうなら、制御はステップS460へ進み、カラード演算子処理が実行される。

【0131】しかしながら、ステップS450において、現在の音韻要素がカラード演算子でないことが決定されると、制御はステップS470へ進み、現在の音韻要素は、それがマスキング演算子であるか否かを知るためにチェックされる。もしそうなら、制御はステップS480へ進み、マスキング演算子処理が実行される。

【0132】しかしながら、ステップS470において、現在の音韻要素がマスキング演算子でないならば、制御はステップS490へ進み、次の音韻要素は、それが幅演算子であるか否かを知るためにチェックされ、もしそうなら、制御はステップS500へ進み、グラフィカル状態演算子処理を実行する。ステップS460、S489及びS500の全てがステップS410へ戻る。

【0133】最後に、現在の音韻要素がステップS490において幅演算子でないことが決定されると、制御はステップS510へ進み、エラー指示がオブジェクト最適化ESS100によって現在の音韻が適切に解析され得ないことを指示するために出力される。次に、ステップS510から制御はステップS520を介してステップS20へ戻る。

【0134】ステップS450でチェックされるカラード演算子が、"setcolor"、"image"、"colorimg"、"e"、及び等価のコマンドのような演算子を含むことが理解されるべきである。勿論、実際のコマンドは、PDL言語に依存する。同様に、ステップS470でチェックされるマスキング演算子コマンドは、"fill"、"stroke"、"character"及び他のこのようなコマンドを含む。また、実際のコマンドは、PDL言語に依存する。最後に、ステップS490でチェックされる状態演算子コマンドは、"setstrokerwidth"、"setrendthint"、及び他のそのような演算子のようなコマンドを含む。また、実際のコマンドが使用されるPDL言語に依存する事が理解されるべきである。

【0135】図13は、図10のステップS240のスキャンラインデータが発生するための処理をより詳細に

示す。図13に示されるように、スキャンラインデータが発生するための処理はステップS600でアクティブオブジェクトリストを初期化しスキャンラインを1にセットすることによって開始する。

【0136】次に、ステップS610において、現在のスキャンラインからの各新たなオブジェクトが登録されたアクティブオブジェクトリストへ併合される。図28は、ステップS210においてオブジェクト指定タグを有するオブジェクトリストの構成の間発生されるスキャンラインオブジェクトリストデータの汎用フォーマムを示す。図28に示されるように、第3のスキャンライン1513上でサートするオブジェクトを指すポインタは、例えば、第3のスキャンライン1513上の第1の(最初の)オブジェクト15131を指し、それは、一方、スキャンライン1513上の第2のオブジェクト15132を指す。この構造において、「15131」、「15132」のようなオブジェクトリストは、それらに生成する対応するPDL音韻要素の相対位置に従ってリストに位置決めされるオブジェクトを含む。

【0137】しかしながら、登録された(ソートされた)アクティブオブジェクトは、オブジェクトの開始位置により且つ左から右の順序で整理される。このように、整理されたアクティブオブジェクトリストの第1のオブジェクトは、最左開始位置を有するアクティブオブジェクトである。同様に、各スキャンライン上の最後のオブジェクトは、最右開始位置を有するオブジェクトである。所与のスキャンラインを処理すると、そのスキャンライン始まる各オブジェクトは、この左から右への順序付けが維持されるように、アクティブリストに併合される。一般に、アクティブオブジェクトリストには、現在のスキャンラインに先立スキャンラインで始まり且つアクティブのままであるオブジェクトを含む。

【0138】ステップS610で新たなオブジェクトが整理されたアクティブオブジェクトリストに併合されると、制御はステップS620へ進む。ステップS620において、現在のスキャンラインに対するランリストが整理されたアクティブオブジェクトリストから発生される。次に、制御はステップS630へ進み、そこで現在のスキャンラインに対するランリストが、スキャンラインの各点に沿って、ランリストのどのランが"頂部"に在るかを決定することによって、即ち、その点であらゆる他のランの下に無いランを決定することによって、平滑化される(flattened)。

【0139】ステップS630において、ランリストが平滑化されると、制御はステップS640へ進む。ステップS640において、現在のスキャンラインに対するコマンド及びカラードの順序付けリストが生成される。コマンドは、適切なカラードがカラードレジスタで利用できるとき、マスキングデータが適切にイネーブル又はディセーブルされること、サンプリングされた画像データが必要

総技術を使用して圧縮される。一旦圧縮されると、マスキングデータは、またRAM部分151のマスキングデータチャンセル部分に格納される。

【0143】次に、ステップS740において、サンプリングされた画像ポインタが抽出及び格納され、且つサンプリングされた画像データが抽出、圧縮及びRAM部分151のサンプリングデータチャンセル部分に格納される。ステップS710-S740で異なるデータ部分及びタイプの全てが一旦抽出、(恐らく)圧縮及び格納されると、制御はステップS750を介してステップS260へ戻る。

【0144】図15は、図12のステップS460の、カラード演算子処理する方法をより詳細に示す。カラード演算子処理は、ステップS800でカラード演算子がサンプリングされた画像を提供するか否かを決定することに よって、開始する。もしそれがサンプリングされた画像でなければ、制御はステップS810へ進む。ステップS810において、カラード演算子が、現在の音韻要素によって指示される一定カラードがカラーパレットに同一のエントリを有するか否かを決定するために同一のエントリを有するか否かを決定する。もしそうならば、制御はステップS820へ進む。そこで現在の次の音韻要素によって指示されるのと同じカラードモード及び画像データが有するパレットエントリが抽出される。

【0145】しかしながら、例えば、図16のステップS1050によって、レンダリングデータが変更されているならば、次に、答えは否定であり、新たなパレットエントリが生成される必要がある。このように、ステップS810の答えが否定であるとき、制御はステップS830へ進み、次に、新たなパレットエントリが、現在の音韻要素によって指示されるカラードモード及び画像データを使用して、カラードパレット内に格納される。次に、ステップS840において、ハッシュング関数(機能)が、適切なパレットインデックスを決定するために、現在の音韻要素によって指示される画像データに適用される。図30に示されるように、パレットは、複数のインデックススロット1521-152nよりなる。パレット1520の各ハッシュ表(テーブル)は、多くのパレットエントリを有し、パレットエントリの各々は、一定カラードエントリ及びサンプリングされたエントリの何れかであり得る。パレットは、後でより詳細に述べられる。次

【0146】次に、ステップS710において、一定カラードデータが、以下で詳述されるように、抽出、圧縮及び格納される。次に、ステップS720において、マスキングデータが抽出、圧縮及び格納される。ステップS730において、マスキングデータが抽出及び格納され、それらのマスキングデータが指すマスキングデータが抽出、圧縮及び格納される。コマンドデータの場合のように、L2のような従来の技術が一定カラード及びマスキングデータを圧縮するために使用される。上述のステップの場合のように、一定カラードデータ、マスキングデータ及びマスキングデータは、平滑化ランリストから抽出され、スキャンラインベースで、夫々RAM部分151の一定カラードデータチャンセル部分に格納される。このマスキングデータは、同様に抽出されて、次にラン長エンコーディング、OCITC group 4及び他の互換システムのような既知の1ビット圧

縮技術を使用して圧縮される。一旦圧縮されると、マスキングデータは、またRAM部分151のマスキングデータチャンセル部分に格納される。

【0143】次に、ステップS740において、サンプリングされた画像ポインタが抽出及び格納され、且つサンプリングされた画像データが抽出、圧縮及びRAM部分151のサンプリングデータチャンセル部分に格納される。ステップS710-S740で異なるデータ部分及びタイプの全てが一旦抽出、(恐らく)圧縮及び格納されると、制御はステップS750を介してステップS260へ戻る。

【0144】図15は、図12のステップS460の、カラード演算子処理する方法をより詳細に示す。カラード演算子処理は、ステップS800でカラード演算子がサンプリングされた画像を提供するか否かを決定することに よって、開始する。もしそれがサンプリングされた画像でなければ、制御はステップS810へ進む。ステップS810において、カラード演算子が、現在の音韻要素によって指示される一定カラードがカラーパレットに同一のエントリを有するか否かを決定するために同一のエントリを有するか否かを決定する。もしそうならば、制御はステップS820へ進む。そこで現在の次の音韻要素によって指示されるのと同じカラードモード及び画像データが有するパレットエントリが抽出される。

【0145】しかしながら、例えば、図16のステップS1050によって、レンダリングデータが変更されているならば、次に、答えは否定であり、新たなパレットエントリが生成される必要がある。このように、ステップS810の答えが否定であるとき、制御はステップS830へ進み、次に、新たなパレットエントリが、現在の音韻要素によって指示されるカラードモード及び画像データを使用して、カラードパレット内に格納される。次に、ステップS840において、ハッシュング関数(機能)が、適切なパレットインデックスを決定するために、現在の音韻要素によって指示される画像データに適用される。図30に示されるように、パレットは、複数のインデックススロット1521-152nよりなる。パレット1520の各ハッシュ表(テーブル)は、多くのパレットエントリを有し、パレットエントリの各々は、一定カラードエントリ及びサンプリングされたエントリの何れかであり得る。パレットは、後でより詳細に述べられる。次

に応じて利用できるとき、メタピットが、圧縮解除、カラード空間変換、トーン再生カーブ処理、IOTコントローラ160及びIOT170によって実行されるレーザ変調セッティング等のハードウェア処理を最適化するために適切にセットアップされることを、確実にするために発生される。パレット参照のフォーマットは、正確なカラード及びレンダリングデータ情報で各命令に対してそのパレットから読み出されることを確実にするために発生される。

【0140】ステップS640の後、制御はステップS650へ進む。そこで消費されたアイテムが整理されたアクティブオブジェクトリストから除去される。現在のスキャンラインはオブジェクトが表れる最後のスキャンラインである時に、オブジェクトが消費される。次に、ステップS660において、スキャンライン数は、現在のスキャンライン数がページの最後のスキャンラインであるか否かを決定するためにチェックされる。もしそうでないならば、制御はステップS670へ進む。そこでスキャンラインの数が1だけインクリメントされる。ステップS670から、制御フローがステップS610へ戻る。しかしながら、現在のスキャンラインが最後のスキャンラインであるならば、制御はステップS690を介してステップS250へ戻る。

【0141】ステップS240でスキャンラインデータが発生した後、制御はステップS250へ進む。図14は、ステップS250をより詳細に示す。処理は、コマンドデータが抽出、圧縮及び格納する事によってステップS700で開始する。コマンドデータの特徴のため、レンベール-ブローウェル(LZW)のような公知の圧縮技術がコマンドデータを圧縮するために使用される。コマンドデータは、平滑化ランリストからスキャンラインベースで抽出される。コマンドデータは、図31に示されるように、RAM部分151のコマンドチャネルデータ部分に格納される。

【0142】次に、ステップS710において、一定カラードデータが、以下で詳述されるように、抽出、圧縮及び格納される。次に、ステップS720において、マスキングデータが抽出、圧縮及び格納される。ステップS730において、マスキングデータが抽出及び格納され、それらのマスキングデータが指すマスキングデータが抽出、圧縮及び格納される。コマンドデータの場合のように、L2のような従来の技術が一定カラード及びマスキングデータを圧縮するために使用される。上述のステップの場合のように、一定カラードデータ、マスキングデータ及びマスキングデータは、平滑化ランリストから抽出され、スキャンラインベースで、夫々RAM部分151の一定カラードデータチャンセル部分に格納される。このマスキングデータは、同様に抽出されて、次にラン長エンコーディング、OCITC group 4及び他の互換システムのような既知の1ビット圧

縮技術を使用して圧縮される。一旦圧縮されると、マスキングデータは、またRAM部分151のマスキングデータチャンセル部分に格納される。

【0143】次に、ステップS740において、サンプリングされた画像ポインタが抽出及び格納され、且つサンプリングされた画像データが抽出、圧縮及びRAM部分151のサンプリングデータチャンセル部分に格納される。ステップS710-S740で異なるデータ部分及びタイプの全てが一旦抽出、(恐らく)圧縮及び格納されると、制御はステップS750を介してステップS260へ戻る。

【0144】図15は、図12のステップS460の、カラード演算子処理する方法をより詳細に示す。カラード演算子処理は、ステップS800でカラード演算子がサンプリングされた画像を提供するか否かを決定することに よって、開始する。もしそれがサンプリングされた画像でなければ、制御はステップS810へ進む。ステップS810において、カラード演算子が、現在の音韻要素によって指示される一定カラードがカラーパレットに同一のエントリを有するか否かを決定するために同一のエントリを有するか否かを決定する。もしそうならば、制御はステップS820へ進む。そこで現在の次の音韻要素によって指示されるのと同じカラードモード及び画像データが有するパレットエントリが抽出される。

【0145】しかしながら、例えば、図16のステップS1050によって、レンダリングデータが変更されているならば、次に、答えは否定であり、新たなパレットエントリが生成される必要がある。このように、ステップS810の答えが否定であるとき、制御はステップS830へ進み、次に、新たなパレットエントリが、現在の音韻要素によって指示されるカラードモード及び画像データを使用して、カラードパレット内に格納される。次に、ステップS840において、ハッシュング関数(機能)が、適切なパレットインデックスを決定するために、現在の音韻要素によって指示される画像データに適用される。図30に示されるように、パレットは、複数のインデックススロット1521-152nよりなる。パレット1520の各ハッシュ表(テーブル)は、多くのパレットエントリを有し、パレットエントリの各々は、一定カラードエントリ及びサンプリングされたエントリの何れかであり得る。パレットは、後でより詳細に述べられる。次

し、ステップS1180に進む。しかしながら、“seetr
enderhint”演算子がステップS1110でセットされ
ると、制御がステップS1130に進み、そこでグラ
フィカル状態における“renderhint”パラメータは、“se
trenderhint”演算子によって指示されるレンダリン
グヒントにセットされる。ステップS1130から制御が
再びステップS1180に進む。

【0160】しかしながら、ステップS1100におい
て、“seclimpep”演算子がセットされると、制御がス
テップS1140へ進み、そこで現在のクリップオーブ
ジェクトが存在し、その完全性属性が“完全な”である
か否かが決定される。もしそうならば、制御がステッ
S1150へ進み、そこで現在の完全なクリップオーブ
ジェクトがスライプオブジェクトへ変換される。次に、
領域r1として示されるスライプアウトラインが、現在
のクリッピング領域にセットされる。同時に、領域r2
で示されるスライプのフィル(充塞)領域が現在のクリ
ッピング領域の下のスライプオブジェクトにセットされ
る。しかしながら、現在のクリップオーブジェクトが存
在しない場合又は現在の既存のクリップオーブジェク
トが“complete (完全な)”の完全性属性を持たない場
合、制御がステップS1140から直接にステップS1
160へ進む。

【0161】ステップS1160において、このオブ
ジェクトが、オブジェクトの最初のスキャンラインとし
て、スキャンライン整列アクティブオブジェクトリス
トに挿入される。ステップS1140からステップS11
60は、図12に示され且つ上述された“クリップレ
ン”処理430を形成する。次に、制御がステップS
1170に進み、そこでグラフィカル領域の“current
lipper”コマンドが新たなクリップ領域にセットされ
る。次に、ステップS1120及びステップS1130
において、制御がステップS1180へ進み、制御がス
テップS410へ戻る。

【0162】図13のステップS620において述べら
れたように、スキャンラインリストを発生するための処
理のより詳細な記述が図18において示される。スキャ
ンラインリストを発生するための処理は、変数“thisob
ject”を現在のスキャンラインのアクティブオブジェク
トリストの最初のオブジェクトへセットすることによ
り、ステップS1200において開始する。

【0163】次に、ステップS1210において、変
数“thisobject”は、それが有効オブジェクトを指すか
否かを決定するためにテストされる。もし変数“thisob
ject”が有効オブジェクトを指さないならば、制御がス
テップS1280を介してステップS630に戻る。し
かしながら、“thisobject”が有効オブジェクトを指す
ならば、“thisobject”が最初のオブジェクトにセット
された直後に常に真となり、制御がステップS1220
に進む。

1020から、制御がステップS1030へ進み、其処
でグラフィカル状態演算子処理によってセットされた
“currentcolor”状態のレンダリングタグが更新される。
図12,15及び16に示されるように、“currentcolo
r”タグがデフォルトタグでなく、“currentcolor”に
よって指示されたパレットエントリに対するレンダリン
グタグと一致しないならば、そのパレットエントリは、
複写されなければならない。その新たなパレットエントリ
のレンダリングタグが更新され、“currentcolor”ボイ
データがその新たなパレットエントリに更新される。

【0156】また、その新たなパレットエントリが処理
カラー演算子ステップS460の間に生成されない場
合、カラーデータは、パレットへ挿入されるのではなく
て、別途保持される。この場合、新たなパレットエント
リは、ステップS1030で生成され、そのカラー値
は、別途保持された所にセットされる。この新たなパレ
ットエントリのレンダリングタグは、ステップS101
0又はステップS1020において決定された値にセッ
トされる。この新たなパレットエントリは、ステップS
1050の部分としてステップS840、S850及び
S910を実行する事によって、この時にパレットへ入
力される。

【0157】次に、ステップS1040において、現在
のオブジェクトは、ボックス、ピットマップ等の一つ以
上のプリミティブ(主)オブジェクトを発生するために
スキャン変換される。スキャン変換は公知の処理であ
る。次に、ステップS1050で、プリミティブオブ
ジェクトの全てが処理されたか否かが決定される。いすれ
かのプリミティブオブジェクトが残っていると、制御が
ステップS1050からステップS1060へ進み、其
処で次の(最初の)プリミティブオブジェクトが現在の
プリミティブオブジェクトとしてマスキング演算子から
得られる。次に、ステップS1070において、プリミ
ティブマスキングオブジェクト処理が現在のプリミティ
ブオブジェクトに対して実行される。次に、制御は、ス
テップS1050へ戻る。このループは、ステップS1
050が処理されるべきプリミティブオブジェクトがな
いことを決定するまで続く。この場合、制御がステッ
S1080を介してステップS410へ戻る。

【0158】図12のステップS500で述べられたグ
ラフィカル状態演算子処理のより詳細なバージョンを示
す。図17において、ステップS500のグラフィカル
状態演算子処理は、“seclimpep”演算子がセットされ
たか否かをステップS1100で決定する事により開始
する。

【0159】“seclimpep”演算子がセットされてい
ないならば、制御がステップS1110に進み、“setren
derhint”演算子がセットされたか否かを決定する。もし
そうでないならば、制御がステップS1120に進
み、この状態演算子に対してグラフィカル状態をセッ
ト

ると、制御がステップS1020へ進み、そこでオブ
ジェクト最適化レンダリングタグが“renderhint”パラメ
ータ(単数又は複数)から導出される。

【0152】即ち、パラメータ“renderhint”がセッ
トされないならば、オブジェクト最適化レンダリングタグ
は、現在の言語要素の決定されたオブジェクトタイプ及
びオブジェクトの異なるタイプ間で提供される区別のレ
ベルの数に依存して、現在の言語要素に対してオブジェ
クトタイプを最初に分析することによって決定される。
区別の単一レベルのみがイネーブルならば、ステッ
S1010において決定されたオブジェクト最適化レンダ
リングタグは、例えば、非画像オブジェクト(例えば、
テキストやラインアートオブジェクト)から画像オブ
ジェクト(例えば、ハードウェアオブジェクト)を区別す
る。もし区別の追加のレベルがイネーブルであったら
ば、追加のレンダリングタグは、同じオブジェクトに異
なるカラー空間変換、異なるトーン再生カーブ、異なる
飽和レベル等が提供されるように、発生される。何れの
場合も、少なくとも一つのレンダリングタグは、そのオ
ブジェクトのデータを専ら且つ時に使用されるレーザパワ
ーセットポイントを示すために各オブジェクト毎に
発生される。

【0153】“renderhint”パラメータがセットされる
と、ステップS1020において、自動的に決定された値
値、そうでなければステップS1010で決定された値
は、ドキュメントクリエータから明示的命令によってオ
ーバライドされる。このように、ドキュメントクリエ
ータは、そうでなければステップS1010でセットさ
れたレンダリングタグをオーバーライドできる。従っ
て、“renderhint”パラメータは、必要でないかもしれ
ないが、オブジェクトタイプが何であるか指定できる。
それは、このオブジェクトタイプがどのオブジェクトタ
グであるかに拘らず、ドキュメントクリエータが観察
者の目を獲得したいと思う物を指示するセントの独立の
タイプであっても良い。或いは、“renderhint”は、こ
のオブジェクトが背景にあり、観察者の目を獲得したく
ないことを指示しても良い。“renderhint”は、オブ
ジェクトのエッジがシャープにされるべきでない事を指示
してもよい。“renderhint”は、このオブジェクトの定
義されたカラーが飽和がブーストされるのではなくて保
持されるべきであることを指示してもよい。“renderhi
nt”は、オブジェクトのそのタイプに対してデフォルト
セットポイントとは異なる特定のレーザパワーセットポ
イントが代わりに使用されるべきであることを指示して
もよい。

【0154】また、“renderhint”は、定義されたオブ
ジェクトタイプ及び未定義のサブパラメータに基づい
て、デフォルトモードに続く分析の残りの部分でオブジ
ェクトタイプを明示的に定義してもよい。
【0155】次に、ステップS1010及びステップS

に、ステップS850において、現在のパレットエント
リが、パレットインデックスによって決定されるハッ
シュテーブルスロットでパレットに挿入される。

【0146】新たなパレットエントリのセットアップの
代わりに、現在の一定カラーは、ステップS1050が
この現在の一定カラーに対してレンダリングタグをセッ
トする事を必要とするまで、別に格納できる。この場
合、この時点でのみ、新たなパレットエントリがデフォ
ルト値からすでにリセットされているレンダリングタ
グフィールドで形成される。

【0147】しかしながら、ステップS800がカラー
演算子がサンプリング画像を提供した事を決定すると、制御
がステップS860へ進む。ステップS860で、画像
解像度係数が画像データ及び利用できる107解像度基
づいて計算される。表4のビット6及び7を参照して述
べられたように、画像解像度係数は、107に対して得る
ことが出来る解像度を発生するために、サンプリングさ
れた画像の各面素が何クロックサイクル繰り返されるか
を指示する。

【0148】次に、ステップS870において、現在の
画像データは、ステップS860で決定された画像解像
度係数及びグラフィカル状態演算子処理によって評価さ
れた最近の“現在変換(currenttransform)”に従っ
て、回転及び/又はスケリングされる。次に、ステッ
S880において、新たなカラーパレットエントリ
は、現在のカラーモデル及び現在のサンプリングされた
画像データを使用して、確立される。

【0149】次に、ステップS890において、パレ
ットインデックスは、ハッシュ関数を画像データハイト
即ち“s-Size (Sサイズ)”へ適用することによって
計算される。次に、ステップS900において、現在の
パレットエントリは、パレットインデックスによって決
定されたハッシュテーブルスロットでカラーパレット
に挿入される。

【0150】次に、ステップS820、S850及びS
900の何れからか、制御がステップS910へ進み、
そこでグラフィカル状態演算子処理で発生された“curr
entcolor (現在のカラー)”ポイントが、ステップS8
20、S850又はS900で決定された現在のパレ
ットエントリにセットされる。次に、制御がステップS9
20を介してステップS410へ進む。

【0151】図16は、図12のステップS480のマ
スキング演算子処理する方法をより詳細に示す。マ
スキング演算子処理は、パラメータ“renderhint (レンダ
リングヒント)”がグラフィカル状態演算子処理によっ
てすでにセットされているか否かを決定することによっ
て、ステップS1000で開始する。もしそうでないな
らば、制御がステップS1010に進み、そこでオブジ
ェクト最適化レンダリングタグが自動的に決定される。
しかしながら、パラメータ“renderhint”がセットされ

【0164】ステップS1220において、“thisobject”は、それがスイープタイプのオブジェクトであるか否かを知るためにチェックされる。もしそうならば、制御がステップS1230に進む。ステップS1230において、現在のスキャンライン上の“thisobject”のs1アウトラインの各ピクセル毎に一つのランが識別される。ランリストに併合される。併合されたランの各々は、そのスキャンラインに付った開始位置及び終了位置からなる。各ランは、また“thisobject”の下層のスイープオブジェクトs2を指すポインタからなり、層及びカラーデータが抽出される。上述のように、s1領域は、スイープが生成された時に有効にクリッピング領域の下層とされ、他方s2領域は現在のクリッピング領域の下層のスイープオブジェクトへセットされた。ステップS1230を実行した後、制御がステップS1270へ進む。

【0165】ステップS1220において、“thisobject”のクラスがスイープでない時、制御がステップS1240へ進み、そこで“thisobject”は、それがクリップタイプオブジェクトであるか否かを知るためにチェックされる。もしそうならば、制御がステップS1250に進む。ステップS1250において、“thisobject”のインサイドラストから各オブジェクトが“thisobject”のクリッパーに抗してクリップされる。インサイドラストは、図23において示され且つ以下でより詳細に述べられる。ステップS1740-S1770のプリミティブマスキングオブジェクト処理の間に、このクリッピングオブジェクトのために、収集されると共にこのリストへ追加されたオブジェクトのリストである。即ち、オブジェクトの各インサイドラストは、現在のクリッピング領域として対応するクリッピングオブジェクトが有効であった面に対応するクリッピングオブジェクトに関連する。“thisobject”のクリッパーは、s1スイープアウトラインに関して上述されたのと同様に、このクリッピングオブジェクト生成された時に有効であったクリッピング領域へセットされた。オブジェクトは、クリッピング領域のクリッパーの外側にあるオブジェクトのあらゆる部分を除去する事によってクリップされる。

【0166】オブジェクトがクリップされた後、各得られたランは、整理されたランリストに併合される。ステップS1220において、各ランはスキャンラインに沿った開始位置と終了位置からなる。しかしながら、ステップS1250において、また各ランは、“thisobject”のインサイドラストから層及びカラーデータへのクリップされたオブジェクトに対するポインタからなる。ステップS1230における場合のように、処理が一旦完了すると、制御がステップS1270へ進む。

【0167】もしステップS1240において、“thisobject”のクラスがクリッパーでない場合、制御がステップS1260に進む。ステップS1260において、

現在のスキャンラインの“thisobject”の各ピクセル毎のランが識別されたランリストへ併合される。ステップS1230及びステップS1250の場合のように、各ランはスキャンラインに付った開始位置及び終了位置からなる。しかしながら、この場合、各ランは、また層及びカラーデータに対する“thisobject”へのポインタからなる。ステップS1230及びステップS1250の場合のように、一旦ステップS1260の処理が終了すると、制御がステップS1270へ進む。

【0168】ステップS1270において、変数“thisobject”が現在の“thisobject”の“次の”フィールドによって指示されるオブジェクトへセットされる。従って、次の“thisobject”が現在の“thisobject”になる。ステップS1270から制御がステップS1210へ進む。

【0169】図19は、図13のステップS360のランリストを平滑化するための処理をより詳細に示す。図19に示されるように、ランリストを平滑化する処理は、ステップS1300で開始する。ステップS1300において、現在のランの前量及び背景オブジェクトは、デフォルメントレンダリングタグを有する白オブジェクトに初期化される。開始位置及び終了位置は、現在のスキャンラインの開始に初期化される。最後に、変数“currentstart”がスキャンラインの開始に初期化される。

【0170】次に、ステップS1310において、変数“currentstart”は、スキャンラインの終了に達したか否かを決定するためにチェックされる。そうでない場合、制御がステップS1320に進み、そこで次の可視ランが識別される。

【0171】次に、制御がステップS1330に進み、そこで次の可視ランの前量及び背景オブジェクトカラーとレンダリングタグは、それらが現在のランのオブジェクトの前量及び背景オブジェクトカラーとレンダリングタグと同じであるか否かを決定するためにチェックされる。これが真ならば、制御がステップS1360に進み、そこでランは、現在のランの終了位置を次の可視ランの終了位置へセットする事によって、結合される。しかしながら、これが真でない場合、制御がステップS1340へ進み、そこで現在のランに対するコマンド及びカラーが生成される。次に、ステップS1350において、次の可視ランが現在のランになる。

【0172】ステップS1350及び“currentstart”がステップS1370に進み、そこで“currentstart”が現在のランの終了位置にセットされる。次に、ステップS1370からの制御がステップS1310に進む。ステップS1310において、“currentstart”がスキャンラインの終了に到達されると、制御がステップS1380に進み、そこでコマンドおよびカラーの最後のセットが現在のランのために生成される。ステップS1380から、制御がステップS1390を介してステップ

い。前景カラーが同じ場合、黒文字ビットマップを有する場合と同様に、二つの隣接する透明画は、論理OR関数をそれらのビットマップに適用することによって結合され得る。この領域のマスキングアルゴリズムの元の書き込みが下のビットに上書きされたので、下のビットマップオブジェクトのビットマップがこのステップでそのマップにORされる。透明背景をその上のオブジェクトにORして結合する事は、上のオブジェクトとは異なる。1”ビットのカラーを使用して透明背景図に到達されるまで解くことができる。この点で、マスキングアルゴリズムの使用におけるコンフリクトが見られた。それは、コンフリクトする透明背景ビットマップの空間的に隣接する”1”ビットのグループをランに変換し”平滑化”処理を再帰的に適用することによって解決される。このように、コンフリクトする透明背景オブジェクトが透明でないオブジェクトのセットに変換される。この処理は、不透明オブジェクト又は層”0”白ペーパーが発見されるまで、背景チェーンに続く。

【0177】図21は、図14のステップS710の一定カラー抽出、圧縮及び絡結処理のより詳細な記述を示す。抽出、圧縮及び絡結一定カラー処理はステップS1500で開始する。ステップS1500において、次の(最初の) カラー参照 (基準) が現在のカラー参照として得られる。現在のカラー参照は、図30に示されるパレットデータ構造のパレットエントリの一つを指すポイントである。

【0178】次に、ステップS1510において、現在のパレットエントリからカラーデータが得られる。図30に示されるように、各一定カラータイプパレットエントリ15230は、このハッシュテーブルスロットの次のパレットエントリを指す次のリンク15231と、このパレットエントリがどのようにレンダリングされるべきかについてのデータを提供するレンダリングタグフィールド15232、及びこれが一定カラーパレットエントリであるか、サンプリングされた画像パレットエントリであるか、他のタイプのパレットエントリであるかを指示するカラークラスフィールド15233を含む。カラーモデルフィールド15234は、画素データがどのタイプのカラーモデル (例えば、RGB、CMYK、CIEXYZ、又はその他) を使用するかを指示すると共に、レンダリングタグフィールドと共にこのカラーをCMYKに変換するための好ましいカラー空間変換を指示する。最後に、画素値フィールド15235は、実際のカラーデータを格納する。

【0179】次に、ステップS1520において、現在のパレットエントリの画素データは、画素値データフィールド15235から読み出される。次に、ステップS1530において、丁度読み出された画素データが、図31に示されるように、一定カラーチャネルの次に利用できる位置でRAM151のチャネルデータ構造に格納

S640へ戻る。

【0173】図20は、図19のステップS1340またはS1380の現在のスキャンラインの現在のランに対するコマンド及びカラーを生成するための処理をより詳細に示す。ステップS1400において、現在のランの前量オブジェクトは、そのオブジェクトが透明か否かを決定するために調べられる。即ち、その照会は、そのオブジェクトが真のクリアフィールド15574を有するビットマップタイプのオブジェクト15570であるか否かということである。もしそうでないならば、制御はステップS1410へ進む。ステップS1410においては、現在のランの前量オブジェクトは、そのオブジェクトがクラススイープタイプのオブジェクト15580であるか否かを知るために調べられる。もしそうでないならば、制御がステップS1420に進み、ノーマルコマンド及びカラーを生成する。次に、制御がステップS1460に進む。

【0174】もしステップS1410において、前量オブジェクトがスイープクラスタイプオブジェクト15590であると決定されると、制御がステップS1430に進み、スイープコマンド及びカラーを生成する。次に、制御が再びステップS1460に進む。しかしながら、ステップS1400において、現在のランの前量オブジェクトが真のクリアフィールドを有するクラススイープタイプのオブジェクトであると決定され、制御がステップS1440に進み、背景チェーンを処理してマスクビットマップを固定する。次に、制御がステップS1450に進み、マスクを使用してコマンド及びカラーを生成する。次に、制御が再びステップS1460に進む。

【0175】この手順は、それが2つのステップに依って呼び出され得る点において一般的ではない。このように、ステップS1460は、ステップS1340及びS1380のどちらががこの手順を呼び出したかを決定して、制御が適切な次のステップ、即ちステップS1350又はS1370、へ戻される。即ち、現在のラン手順に対してコマンド及びカラーの生成がステップS1340で呼び出されると、制御がステップS1350に進む。同様に、現在のラン手順に対してコマンド及びカラーの生成がステップS1380で呼び出されると、制御がステップS1390に進み、制御をステップS640へ戻す。

【0176】上述のように、ステップS1400は、現在のランの前量オブジェクトが透明 (即ち、ビットマップブラスオブジェクト) の場合、その背景フィールドによって指示されるオブジェクトがこのランのマスクの”0”ビットのカラーを決定すると共に、現在のランのオブジェクトが前景カラーを決定するように動作する。しかしながら、背景オブジェクトそれ自体は、同じ又は異なる前景カラーを有する透明ビットマップであってもよ

ルームを生成する。この処理のきわめて重要な部分は、ペーパーの左端で始まる。この処理において可視のランの最も多い次の部分を識別することである。これは、ステップS1320の次の可視セグメント識別(Identify Next Visible Segment)処理によって達成される。その基本的アプローチは単純である。ランがそれらの開始位置によって整理されるので、それらは、"アクティブランリスト"に追加されることで、ランのリストは、スキャンラインに沿うこの点でアクティブであると現在考えられている。ランはスキャンラインに沿ってそれらの開始位置によって整理されるので、それらは、それらの開始位置に到達した時にアクティブランリストに追加されることで、スキャンラインに沿う位置がそれらの終了位置を超えて前進した時に破棄され得る。次に、最も高いターゲッド値を有するランが、"頂部(on-top)"として識別される。しかしながら、二つのキーファクタ、性能ファクタ及び透明ビットマップオブジェクトは、処理を複雑にする。

[0200] 幾つかのグラフィカル構造は、この単純なタイプの処理を困難にするPDLにおいて隠されている。一例は、放射状スイープとして知られているものである。放射状スイープは、最下層の大きなオブジェクトと、より下の層上の漸進的により小さいオブジェクトとよりなり、あらゆるより上のオブジェクトは、そのようにより下のオブジェクトの全ての内側に収められるべき性質に入る。ページ上の各このようなスイープは、数百の層を持つことができる。一つの問題は、オブジェクトのスタックの頂部又はその近くに、アクティブランリスト中に見られる非常に多くのオブジェクトがある。ここで表される例示の実施の形態は、図24のステップS1830の直下チェーン化処理を導入する事によって略完全に軽減する。直下チェーン化は、それがより高い層によって覆い隠される時に、現在のより高いランが終了した後、より低い層のランが再出現する限り、それを覆い隠すランへリンクする事によってアクティブランリストから現在の最も高い層を除去する。もしより高いランの前により低いランが終了する場合、より低いランは、それが覆い隠される点で完全に破棄され得る。このようには、一時的に覆い隠されたオブジェクトは、アクティブランリストに配される代わりに、チェーン化されるので、アクティブランリストは非常に短く維持される。このように、放射状スイープの全てのランは、現在の頂部ランの直下の長いチェーン内にある。より上のランが終了すると、そのランの直下のチェーンの最初のランがアクティブランリストへ追加される。新たなランは、その点で頂部でない整理されたランリストに遭遇すると、それは、適切な層で直下のチェーンに追加される。即ち、それは、自身よりも下の層を有するランの上方であって且つより高い層を有するランの下方向へ追加される。更に、ランは、直下のチェーンから除去され、且つそれら

中のエントリへのポインティングである。参照されたパレットエントリは、一定カラーデータやサンプリングされた画像データへ制限されず、図15に示されるカラー演算子処理ステップS460の結果として生成されるパレットエントリであってよい。オブジェクトクラス指定手順フィールド15540は、一つのオブジェクトクラスから他のオブジェクトクラスへ手順の詳細な動作において変化する手順の収束へのポインタである。このように、同じタイプ又はクラスを有する全てのオブジェクトのオブジェクトクラス指定手順フィールド15540は、同じ手順を示す。指定オブジェクトクラスの手順は、そのオブジェクトクラス指定のデータへ正確にアクセスできる。

[0205] 図29にも示される、ボックスリストクラスデータフィールド15560に対するオブジェクト指定クラスデータフィールド15561は、オブジェクトを特定クラスデータフィールド15561及び現在のボックスポインタフィールド15561とから成る。リンクされたボックスポインタフィールド15562から成る。リンクされたボックスポインタフィールド15562は、この一連のボックスの各ボックスは、この一連のボックスの次のボックスへのリンク及びこのボックスの左下及び右上コーナーの位置より成る。従来公知の幾つかの技術は、このような一連のボックスに表され得るボックスへ有用な制御を配する。幾つかの制約は、このような連なるボックスによって述べられるクリッピング領域に作用するクリッピング手順の性能を増加するのに有用である。このように、ボックスリストクラス15562の現在のボックスポインタフィールド15562は、クリッピング及び他の手順に便利なものとして提供される。

[0206] ビットマップ指定クラスデータフィールド15570は、ビットマップオブジェクトポインタフィールド15571を含み、このビットマップは、その自身の境界ボックスデータを有する。ビットマップに1の値を有するデータビットは、そのビットで表される点におけるオブジェクトがパレットエントリポインタ15530によって参照されるカラーでプリントされるべきであると言ふことを指示する。0の値を有するデータビットは、クリアフィールド15574の値に依存して、二つの交互する意味の一方をとる。クリアフィールド15574が値を表す0の場合、マップの0のデータビットは白を表す。クリアフィールド15574が真を表す非0である場合、マップの0のデータビットは、ビットマップがそれらの点で透明であり、カラーがビットマップオブジェクト下でオブジェクトによって決定されることを表す。

[0207] アウトラインオブジェクトポインタフィールド15572は、ビットマップオブジェクト15570のアウトライン又は境界を表す。参照されるアウトラインオブジェクトは、一般にボックスリストクラスオブ

ジェクトである。このように、複雑な形状が表されることで、ビットマップ15770自体が矩形であることが可能である。背景オブジェクトポインタフィールド15573は、クリアフィールド15574によって指示されるように、ビットマップが透明である場合、平滑化処理の間使用される。

[0208] クリップ指定クラスオブジェクトデータフィールド15580は、クリッパーオブジェクトポインタ15581、オブジェクトインサイドクリッパー領域ポインタ15582及び完全性属性15583を含む。クリッパーオブジェクトポインタ15581は、図17の"seclip"演算子ステップS1170によってセッティングされるように、クリッパー領域の形状を指定するためにボックスリストクラスオブジェクトを指す。オブジェクトインサイドポインタ15582は、このクリップクラスオブジェクトによって表されるクリッパー領域が現在のクリッピング領域である時、図23に示されるようなクリッピング領域を表す現在のクリップクラスオブジェクトがクリッピング領域の境界ボックスを充填(フィル)しながら部分的に完全なスイープがいかに完全にどのような方法で収集されるべきかをエンコードするために使用される。

[0209] スイープ指定オブジェクトクラスデータフィールド15590は、s1オブジェクトポインタフィールド15591、s2オブジェクトポインタフィールド15592、スイープ変換レートフィールド15593、及び着色ランプロジェクタポインタフィールド15594より成る。s1オブジェクトポインタフィールド15591は、スイープのアウトラインオブジェクトを指すために主に使用され、ボックスリストクラスオブジェクトは、スイープからなるオブジェクトが収集されていた時に有効であるクリッピング領域を表す。s2オブジェクトポインタフィールド15592は、クラススイープの第2のオブジェクトを指し、そのs1及びs2オブジェクトは、スイープの変化による着色されたスライスを表す単純なオブジェクトのリンクされたリストの二つの端を指す。スイープの変化レートフィールド15593は、図17に示されるように、グラフィカル状態演算子処理ステップS1150に示されるように、クリッピング領域のインサイドでオブジェクトを収集するために使用されるクリップクラスオブジェクトがスイープクラスオブジェクトへ交換される時に計算される。それは、上述の動作の"スローサンプリングチャネル"モードの使用を保証するのに十分な頻度でスイープが変化するカラーであるか否かを決定するために使用される。

[0210] 着色(カラー)ランプロジェクタ方法15594は、収集されたスイープの固有の手順であり、垂直に変化するスイープ、水平に変化するスイー

ブ、及び種々の他の特徴を有するスイープは、異なるカラーランプリジェクション方法手順である。この手順は、図20に示される生成スイープコマンド及びカラーステップS1430の間に呼び出され、このフィールド15594が指すことが出来る各青色スライスを、コマンド及びカラーを生成する。

【0211】図25は、図20のステップS1420のノーマルコマンド及びカラー生成処理のより詳細な説明を示す。ノーマルコマンド及びカラー生成処理は、ステップS1900で開始し、そこでオブジェクトのカラークラスは、そのオブジェクトがそのカラーとしてサンプリングされた画像を有するか否かを決定するためにチェックされる。もし否ならば、制御がステップS1910へ進み、このステップでカラーレジスタの一つにこのランに対する適切なカラーデーター及びメタビット値がロードされることを確保する。このランの前景に対するパレット参照がカラー/メタビットレジスタ0-2に対するシャドールレジスタの各々に保持されているパレット参照と比較されて、オブジェクトによって参照されるカラーがカラーレジスタの一つに以前にロードされていたか否かを決定する。これがそうでない場合、カラー/メタビットレジスタ0-2の一つが選択されて以下のステップで発生されるコマンドによってカラーデーターがロードされる。この選択は、“最低使用頻度 (least recently used)” 処理や他の同義又は等価処理のようならゆる従来の処理を使用できる。最後に、このランに対応するパレットに対する参照は、一定カラーチャネルデーター構造の次に利用できる位置へ出力される。

【0212】次に、制御がステップS1920へ進み、そこでカラーレジスタ選択値、ロードカラー及びこのカラーレジスタ選択値によって選択された特定のカラーレジスタに対する適切なビット値を有するノーマルコマンドが、ステップS1910で決定された値に基づいて出される。更に、多くの“繰り返し”コマンドは、必要に応じて、ノーマルコマンドにおける64画素長制限を超えてランを拡張するために発生される。ステップS1920から、制御がステップS1960へ進み、そこで制御がステップS1460へ戻される。

【0213】もし、ステップS1900において、オブジェクトのカラーがサンプリングされた画像である場合、制御がステップS1930へ進み、それによりサンプリング画像に対応するメタビット値は、画像画素が表示される時にメタビットレジスタ3へロードされることが確実になる。シャドールレジスタは、全体のコマンド及びカラーパレット参照発生処理の間メモリ150のRAM部分151に保持され、それによってカラーレジスタの現在の内容が決定できる。この場合、カラーレジスタ3の値は重要ではない。メタビットレジスタ3の値のみが使用され、カラー画素データーがサンプリングチャネルによって供給される。シャドールレジスタの内容に基づいて、メタビッ

の間に計算される。このフィールドは、スイープにおけるカラーステップ毎に使用される画素の平均数を指示する。変化レートがカラーステップ当り2画素以下である場合、制御がステップS2020へ進み、カラーレジスタ選択値を使用してカラー/メタビットレジスタ3を選択する一つの単一コマンドを発生する。上述のように、カラーレジスタ選択値を有するカラー/メタビットレジスタ3を使用する時、一つのカラーがクロックサイクル毎に一定カラーチャネルから読み込まれる。このスローサンプリングチャネルを使用する事によって、単一のコマンドのみが、各カラースライスを毎に唯一つのカラーパレット参照と共に、要求される、100セントロワードセットアップするコマンドビット及びカラーパレット参照は、コマンドチャネルメモリ及びび一定カラーチャネルメモリの次に利用可能な位置へ出力される。

【0218】次に、制御がステップS2020からステップS2030を介してステップS1460へ進む。【0219】しかしながら、スイープの変化レートが2画素を超えると、制御がステップS2010へ進む。ステップS2010において、コマンド及びカラーパレット参照が発生されてコマンドチャネルメモリ及びび一定カラーチャネルメモリへロードされなければならない。ステップS2020の場合のように、制御がステップS1460へ進む。

【0220】図27は、ステップS1450のマスク処理を使用するコマンド及びびカラー生成のより詳細な記述を示す。この処理は、コンパイナがノーマルにラン(実行)することを可能とする条件を確立するために、最上1のコマンドが発行しなければならない特別な場合は、透明スイープの両方のカラーをロードすることの必要性、又はマスクを使用して画像の前景と一定カラー前景の間のスイッチをする時に、メタビット値及びび一定カラーの両方をロードすることの必要性を含む。これらの場合の幾つかにおいて、マスクデーター自体の最初(第1)のビットカラーレジスタがロードされる順序を決定するために調べられなければならない。それによってマスクデーターの最初の画素によって選択されるカラーがその画素を出すコマンドによってロードされたカラーレジスタを選択する。

【0225】次に、制御がステップS2170へ進み、そこで2番目にロードされたと決定されたカラーがその二つカラーの間で適切に選択するマスクを有するランの残りに対するコマンドを発生する事によってロードされる。このように、第2番目のコマンドは、コマンドチャネルメモリの次に利用できる位置へロードされ、且つ第2番目のパレット参照が一定カラーチャネルの次に利用できる位置へロードされる。

【0226】しかしながら、ステップS2130で前景及び背景カラーの両方がロードされる必要がある場合、制御がステップS2160へ進み、そこで2番目にロードされるマスクを選択しなければならない。この場合において本例のカラーレジスタ選択値によって選択されなければな

らない一定カラーを前景に配置するために指定されなければならない。

【0221】その処理は、ステップS2100において開始し、そこで前景及び背景カラーは、それらの何れがサンプリングされた画像カラーであるかを知るためにチェックされる。肯定の場合、制御がステップS2110へ進み、そこで画像メタビット及びび一定カラーがそのマスクのために検証される。即ち、カラーレジスタ及びメタビットのロード順序は、その処理が正確に動作することを確保するために決定される。この決定は、カラーレジスタの何れかに必要とされる一定カラーがあるかないかを決定し、且つメタビットレジスタ3のメタビット値が明示的にロードされる必要があるか否かを決定することによってマスクデーターの初期ビットを問合せることによってなされる。次に、ハードウェアを適切な状態に適切な初期化するためにここでも必要とされるように決定される1画素コマンドの数は、引き継ぐステップでコマンドを発生するために使用される。

【0222】次に、制御がステップS2110からステップS2120へ進み、ステップS2110で決定されたデーターを実際に発生する。

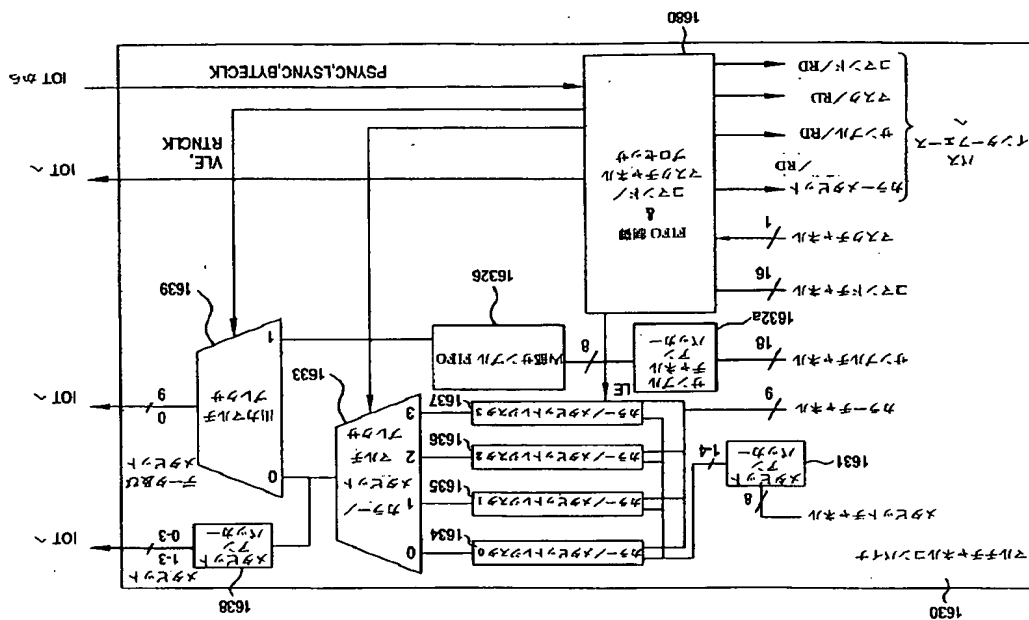
【0223】次に、制御がステップS2120からステップS2190を介してステップS1460へ戻る。

【0224】しかしながら、若し、ステップS2100において、前景及び背景がサンプリングされた画像カラーでないならば、制御がステップS2130へ進み、そこでシャドールレジスタは、前最及び背景カラーの両方がロードされる必要があるか否かを決定するためにチェックされる。若しそうならば、制御がステップS2130からステップS2140へ進み、ロードされる必要のある二つのカラーの内どちらが最初にロードされるべきかを決定する。また、上述のように、これは、マスクデーターの最初のビットを調べることによってなされる。最初にロードされる必要があるカラーがステップS2140で決定されると、制御がステップS2150へ進み、そこで1画素長コマンドが発生されて、最初のカラーをロードするためにコマンドチャネルメモリの最初の利用できる位置へロードされる。勿論、最初のカラーは、一定カラーチャネルの次に利用できる位置へロードされる。

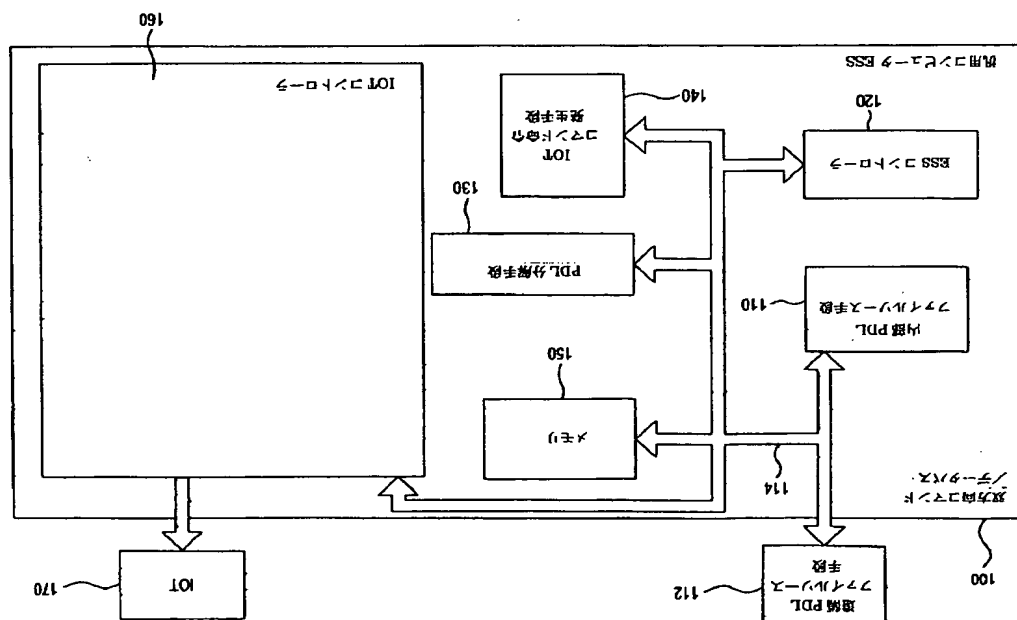
【0225】次に、制御がステップS2170へ進み、そこで2番目にロードされたと決定されたカラーがその二つカラーの間で適切に選択するマスクを有するランの残りに対するコマンドを発生する事によってロードされる。このように、第2番目のコマンドは、コマンドチャネルメモリの次に利用できる位置へロードされ、且つ第2番目のパレット参照が一定カラーチャネルの次に利用できる位置へロードされる。

【0226】しかしながら、ステップS2130で前景及び背景カラーの両方がロードされる必要がある場合、制御がステップS2160へ進み、そこで2番目にロードされるマスクを選択しなければならない。この場合において本例のカラーレジスタ選択値によって選択されなければな

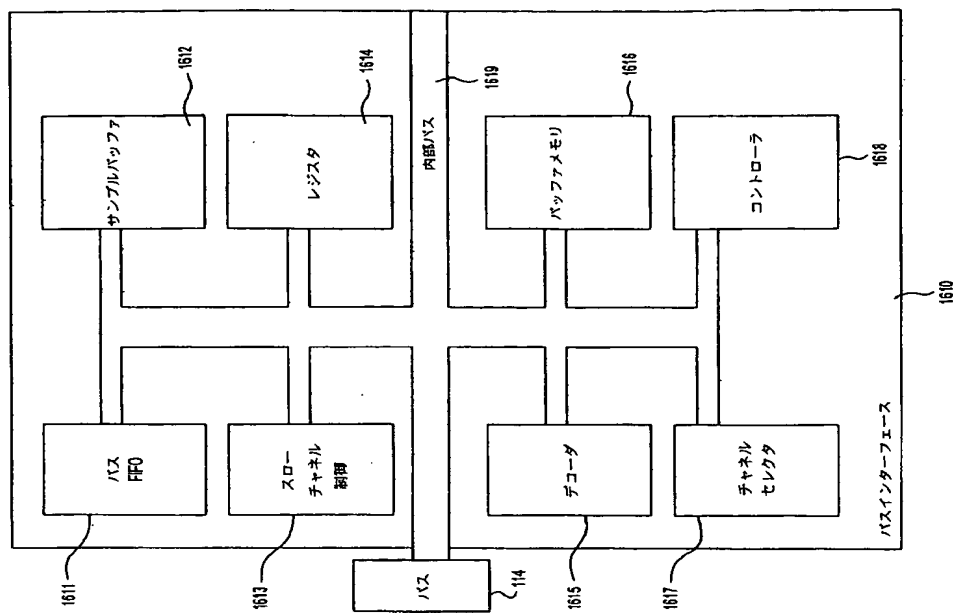
【図3】



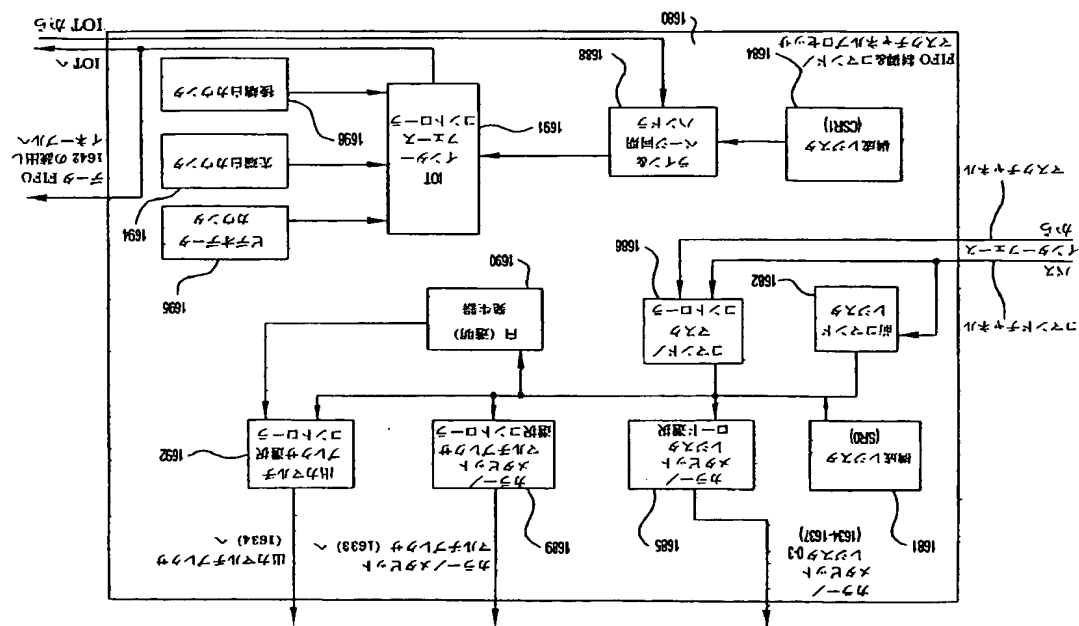
【図1】



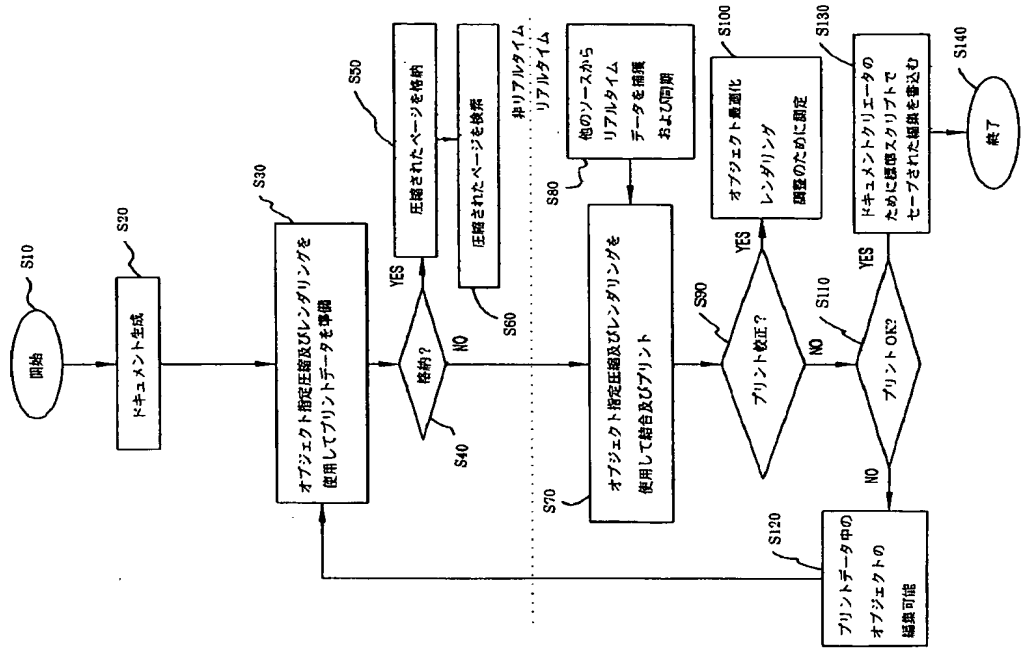
【図 5】



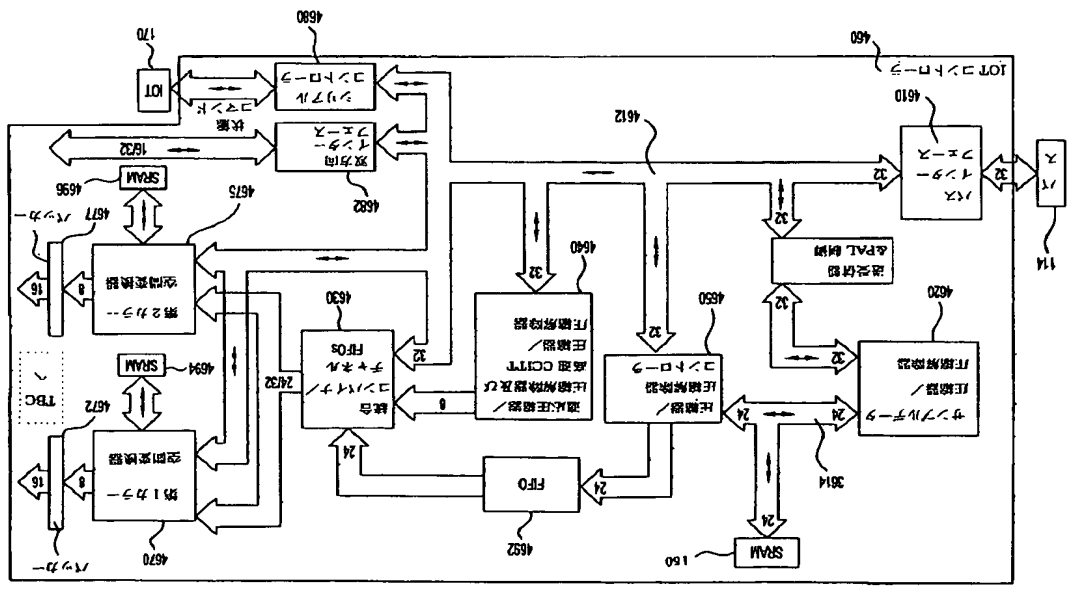
【図 4】



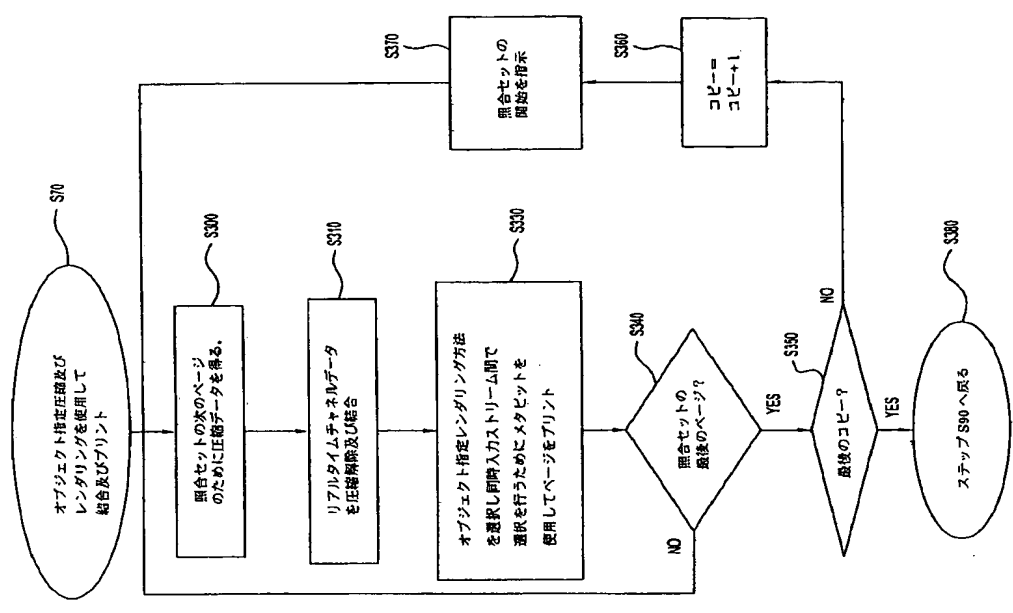
【図 9】



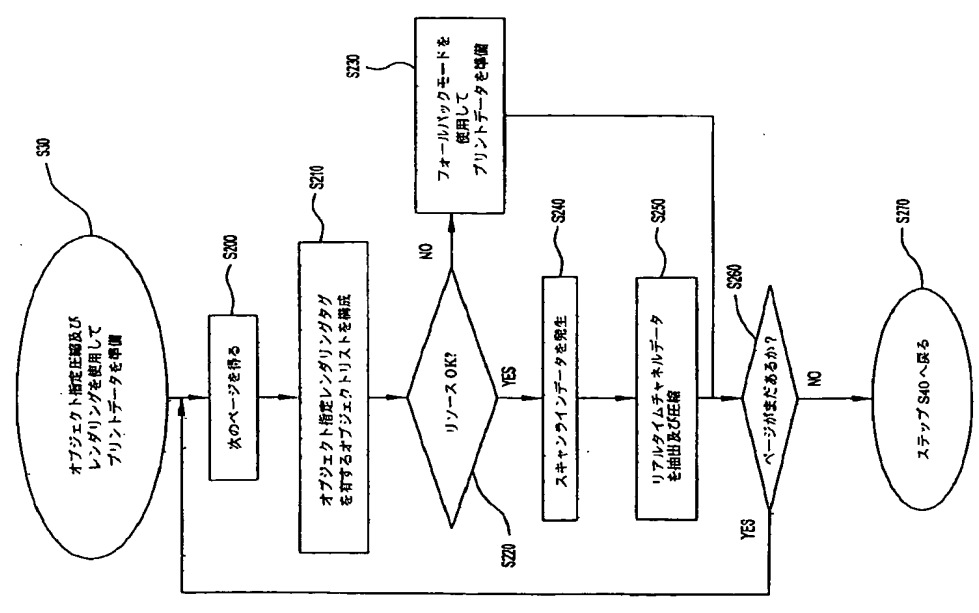
【図 8】



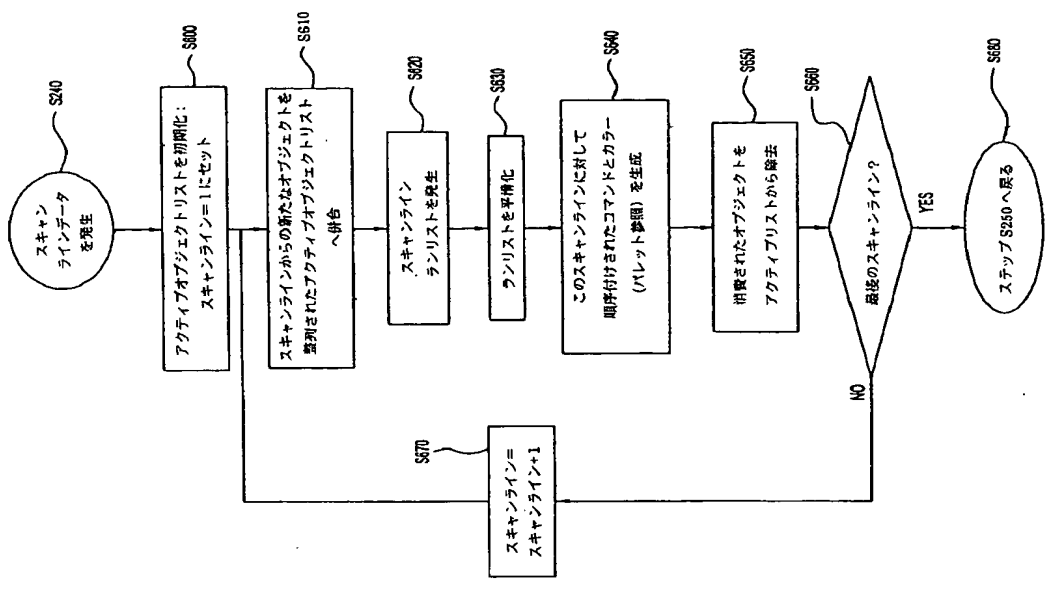
【図11】



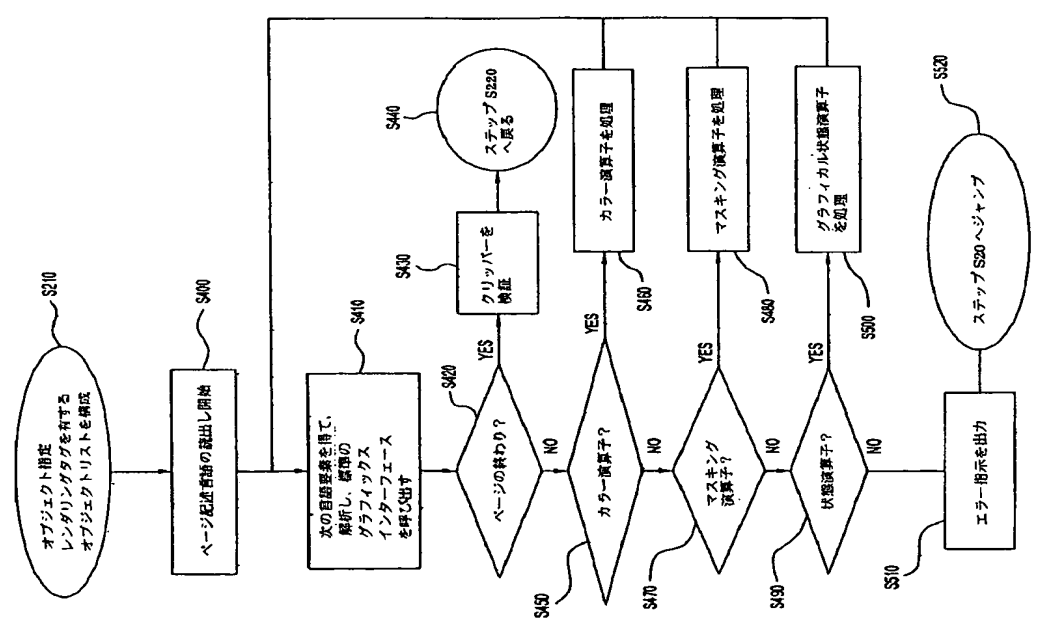
【図10】



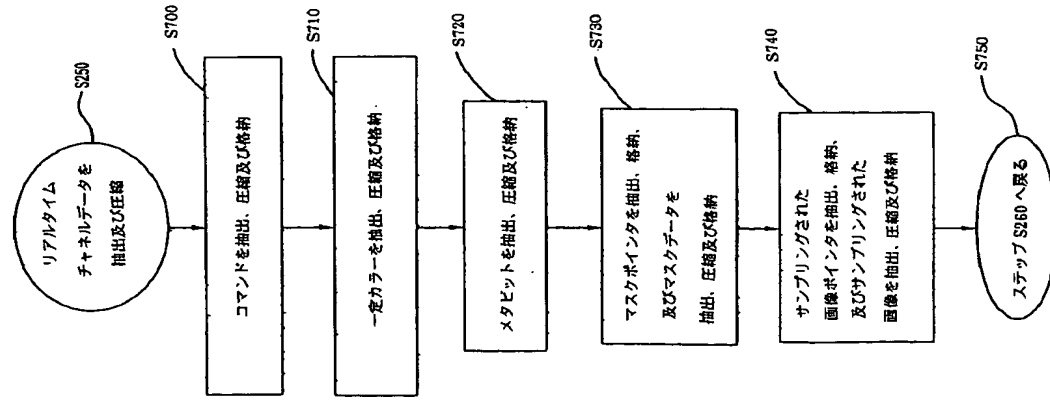
【図13】



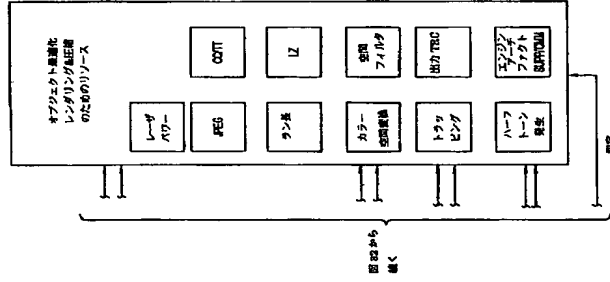
【図12】



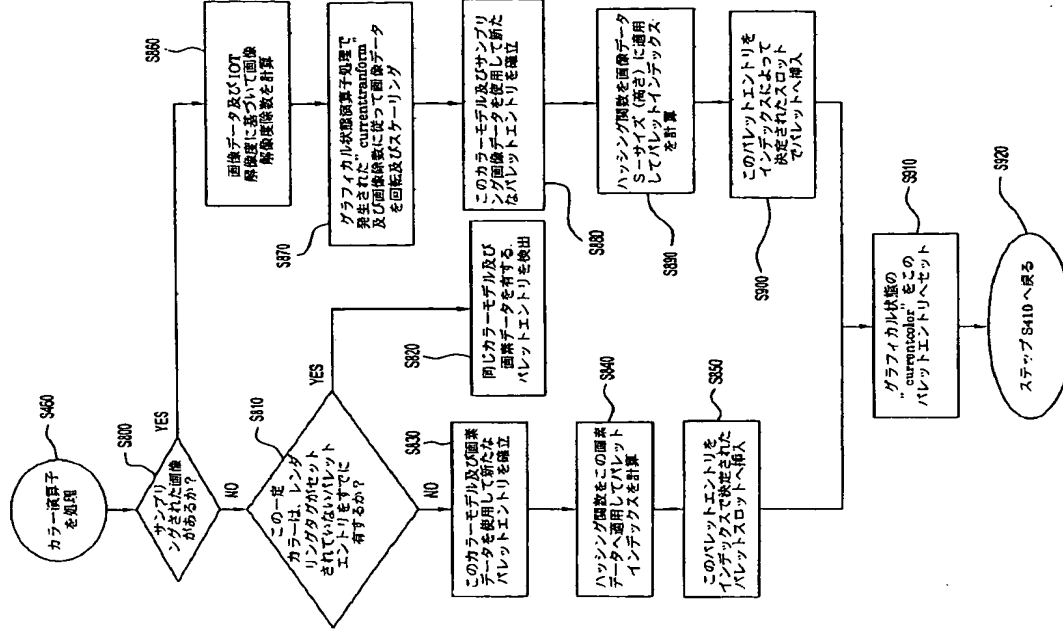
【图14】



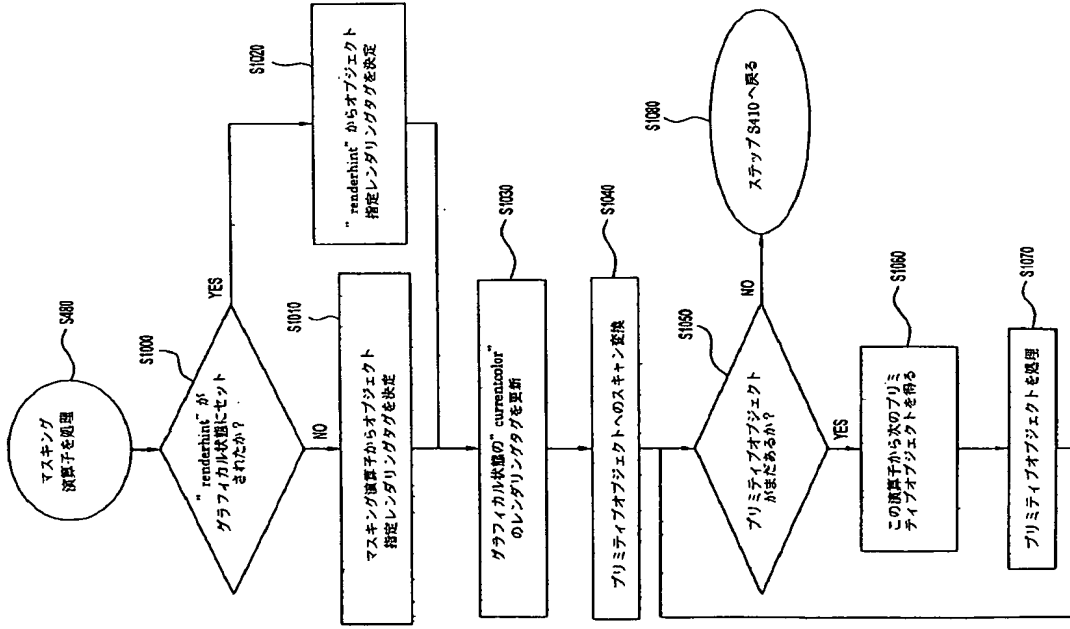
【33】



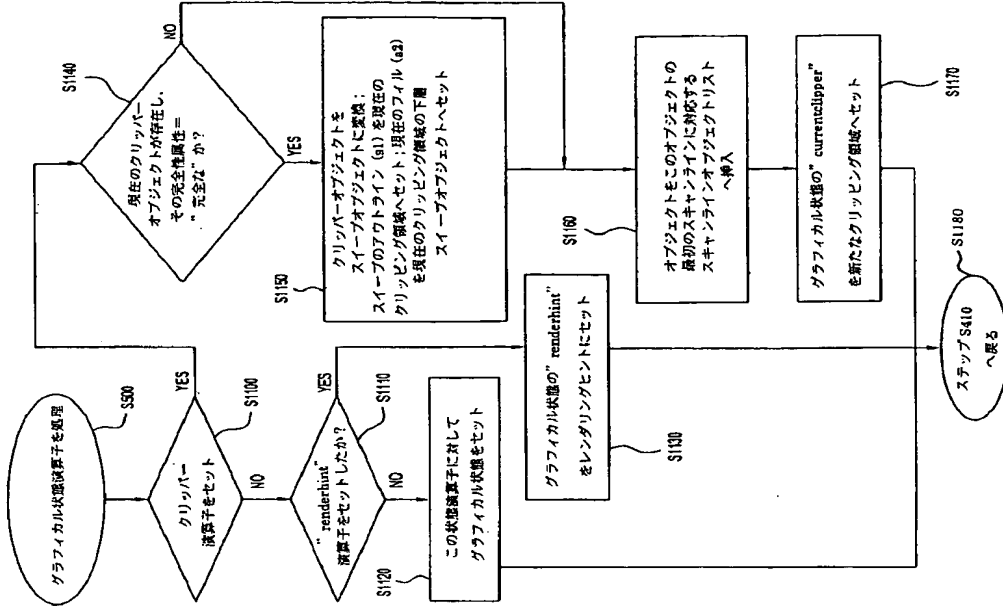
【图15】



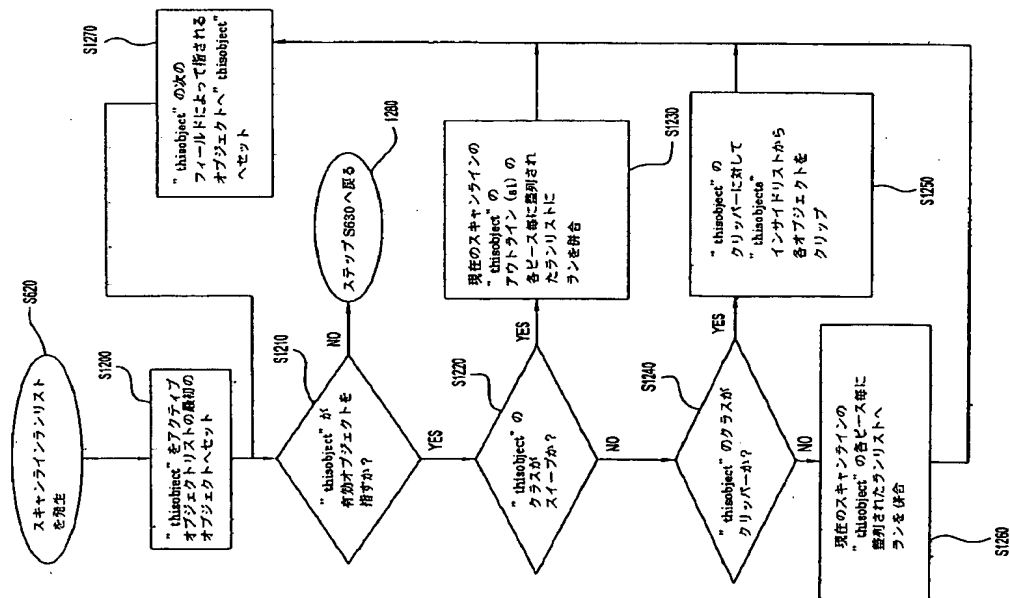
【図16】



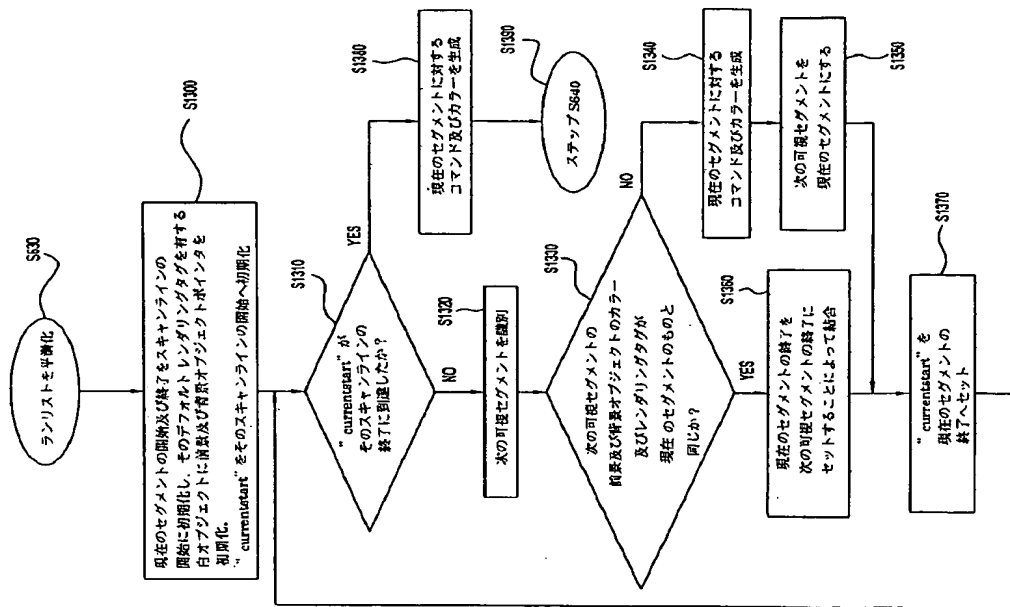
【図17】



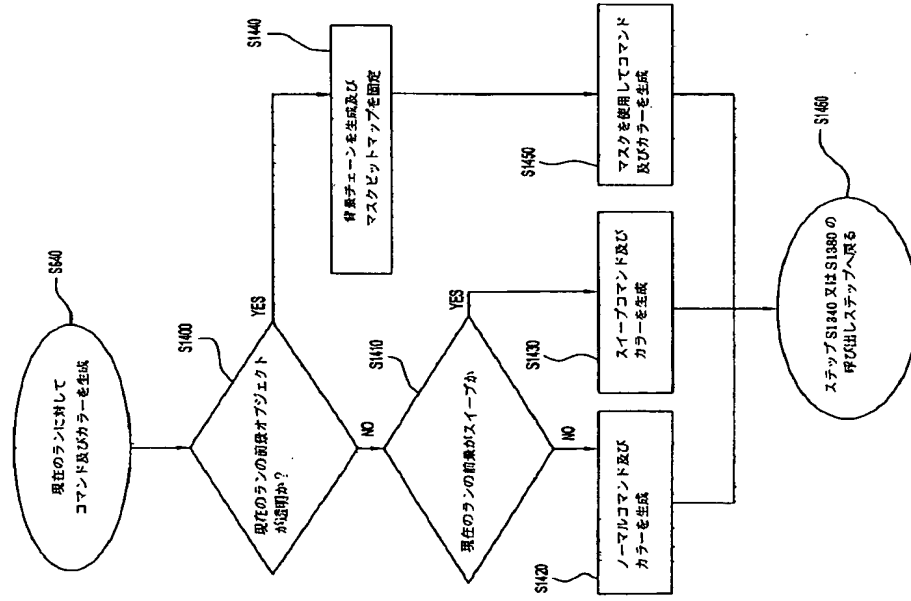
【図 18】



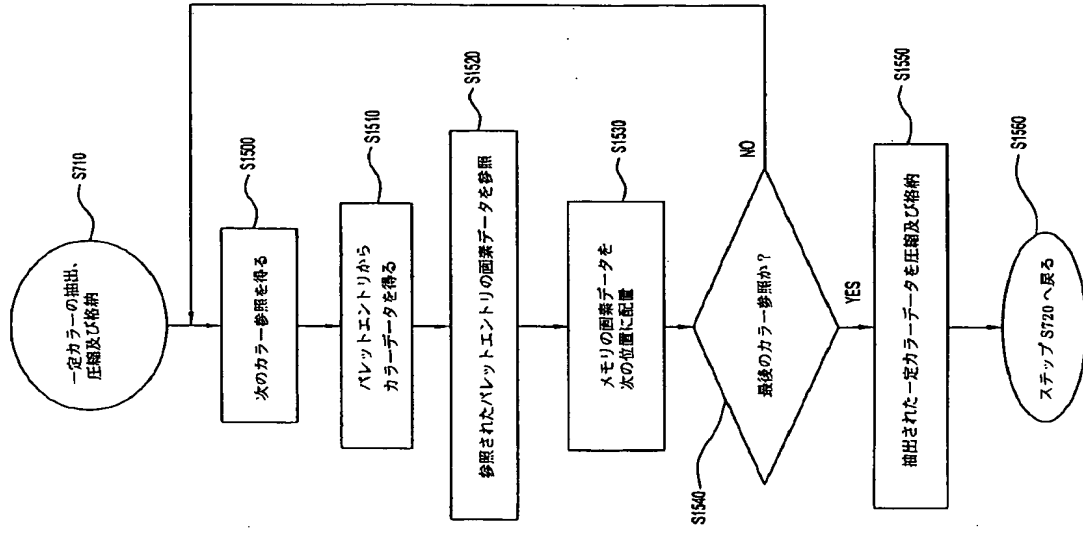
【図 19】



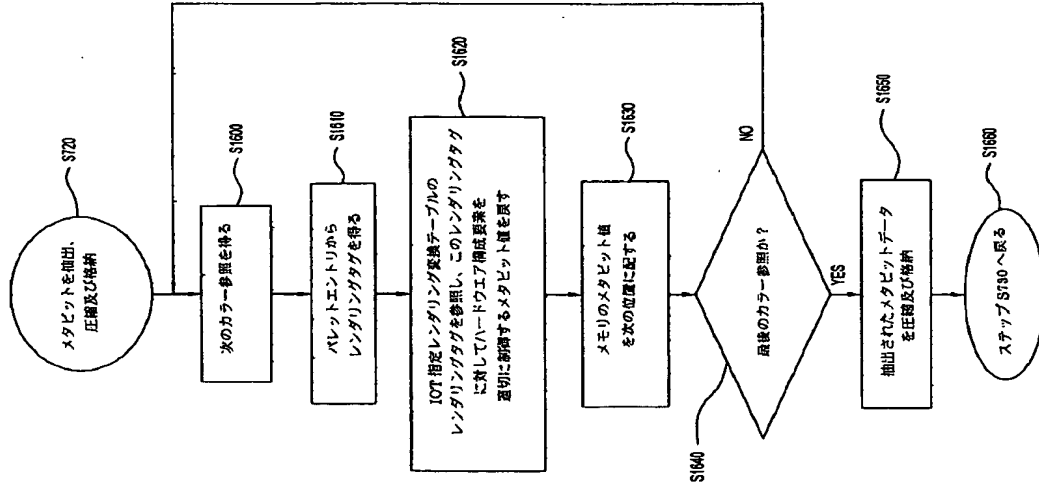
【図 20】



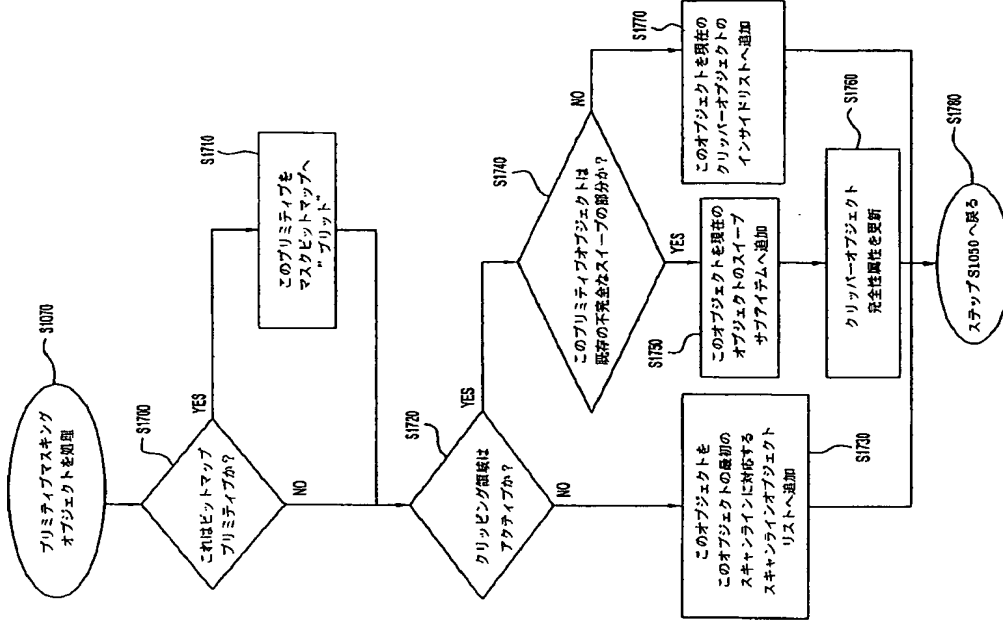
【図 21】



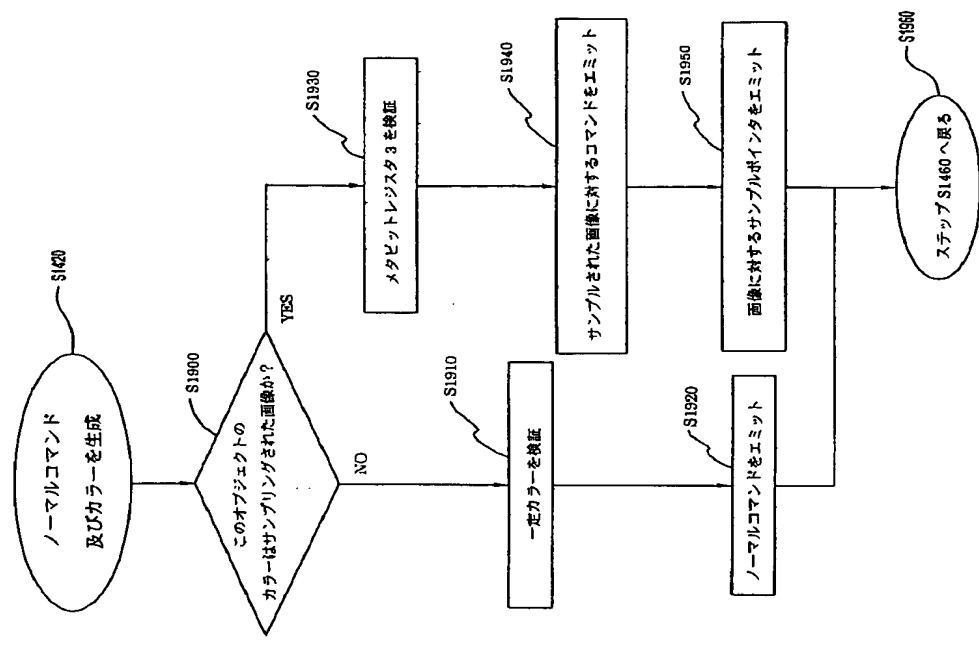
【図 22】



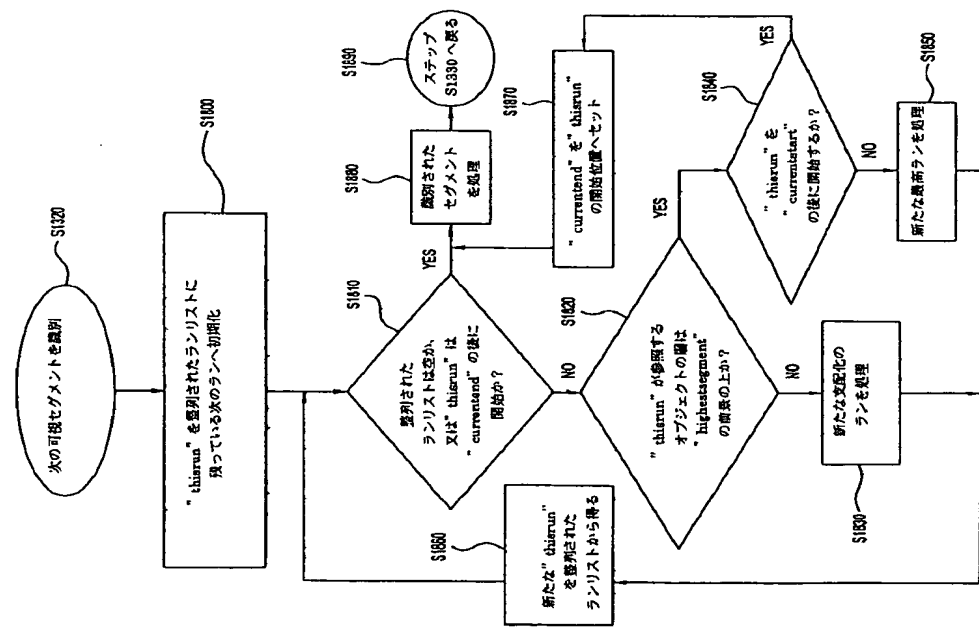
【図 23】



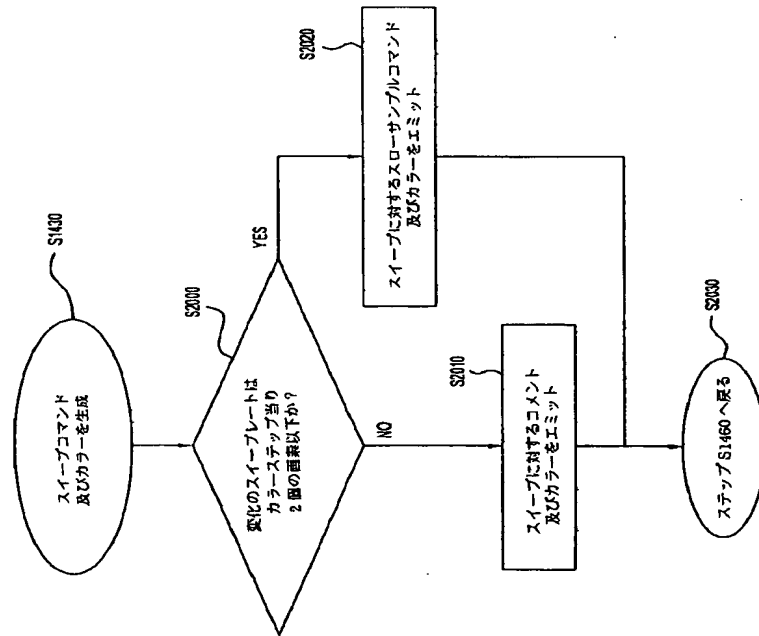
【図 25】



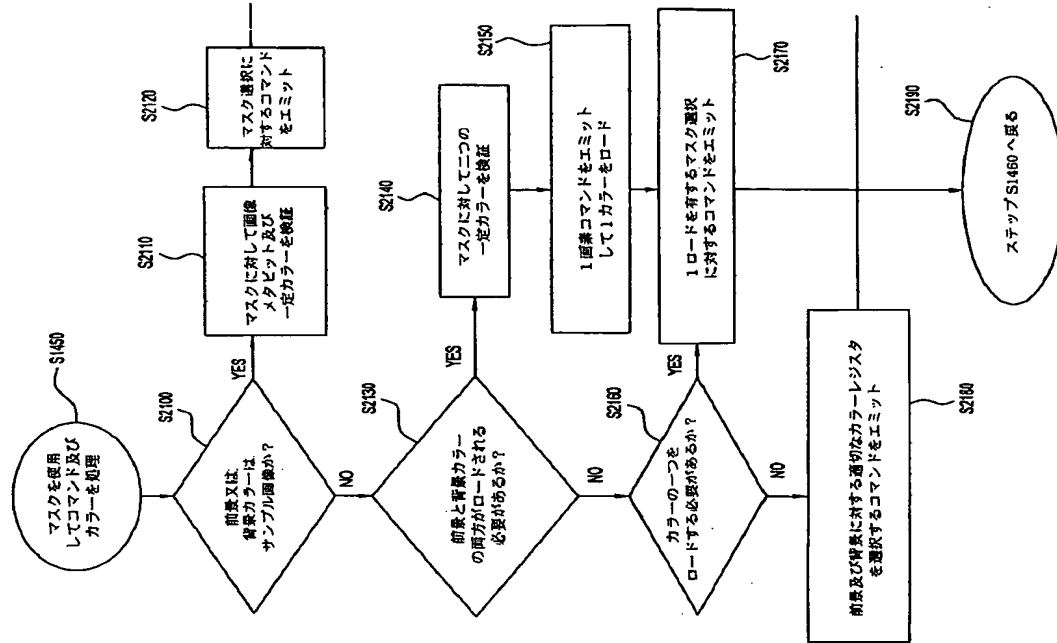
【図 24】



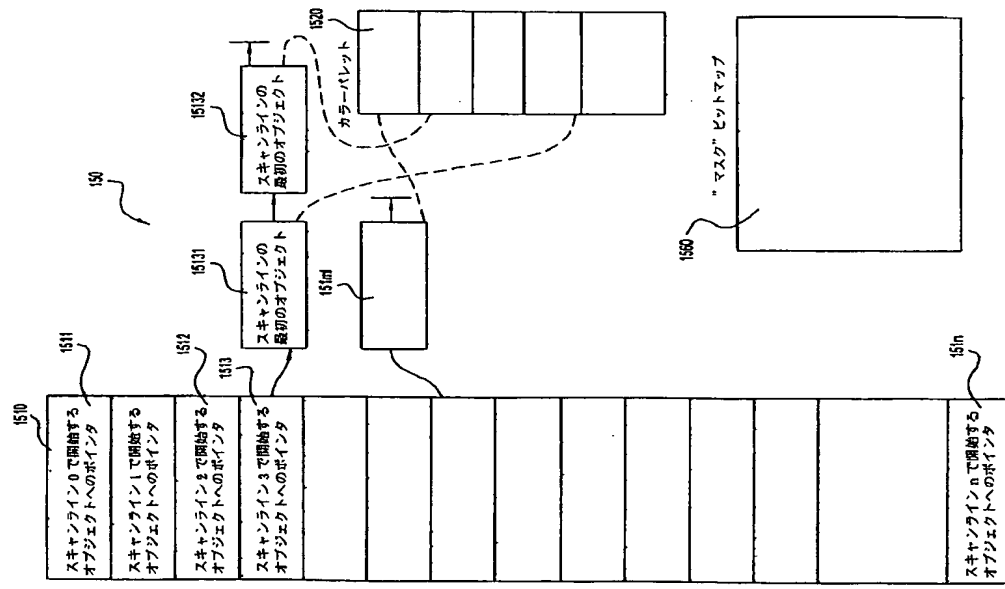
【图26】



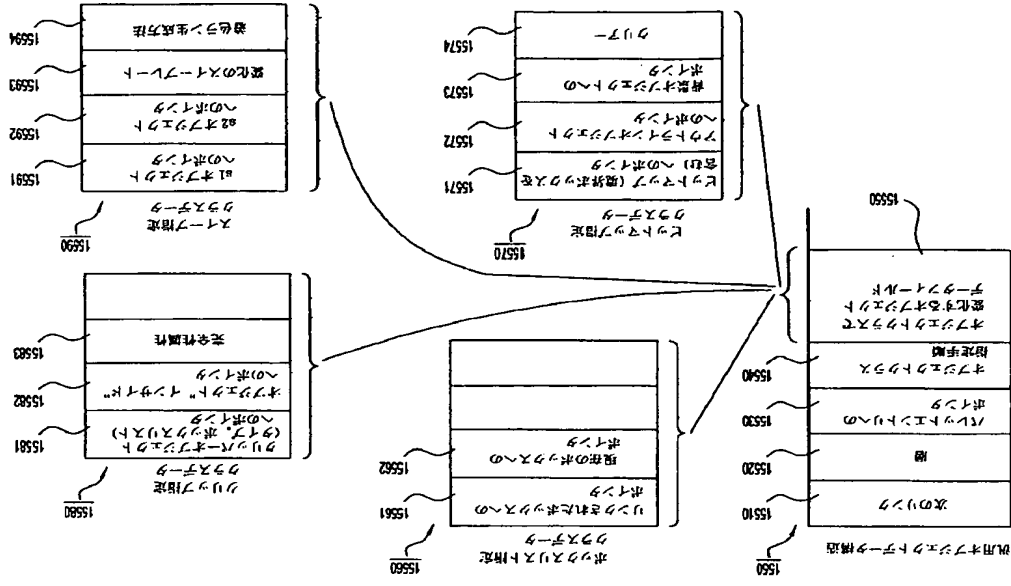
【图27】



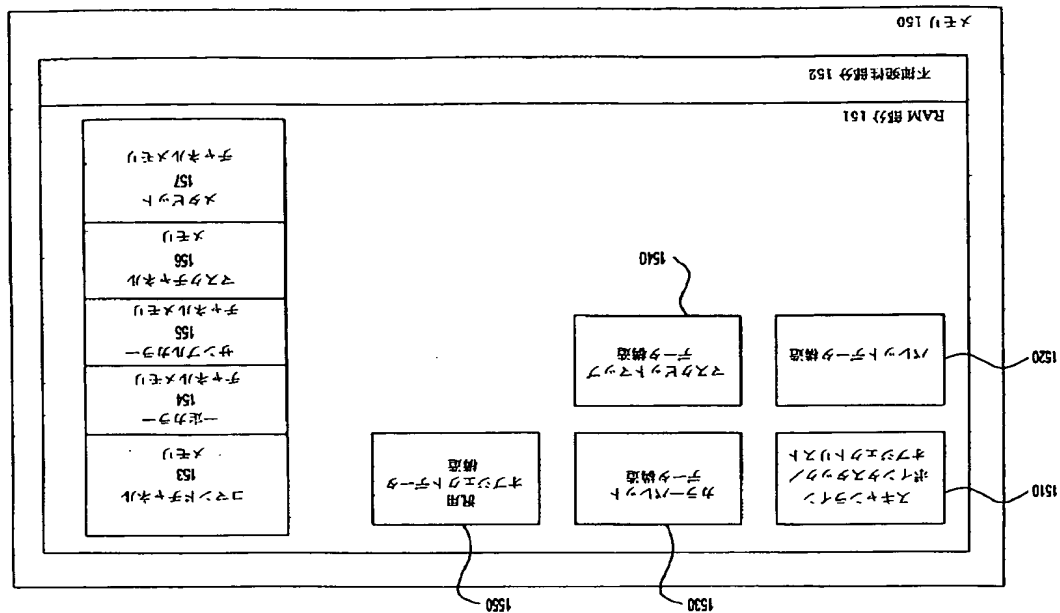
【28】



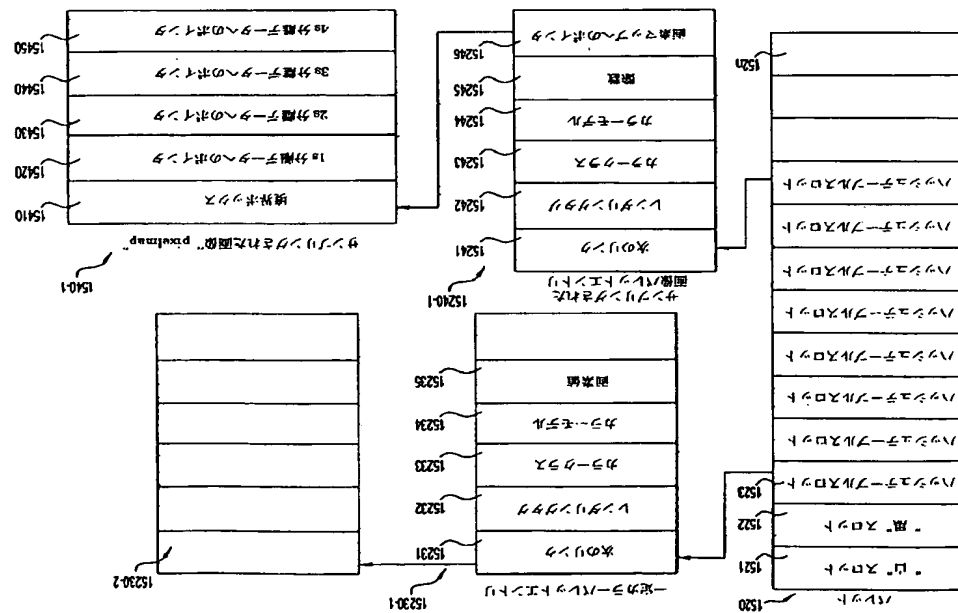
【図29】



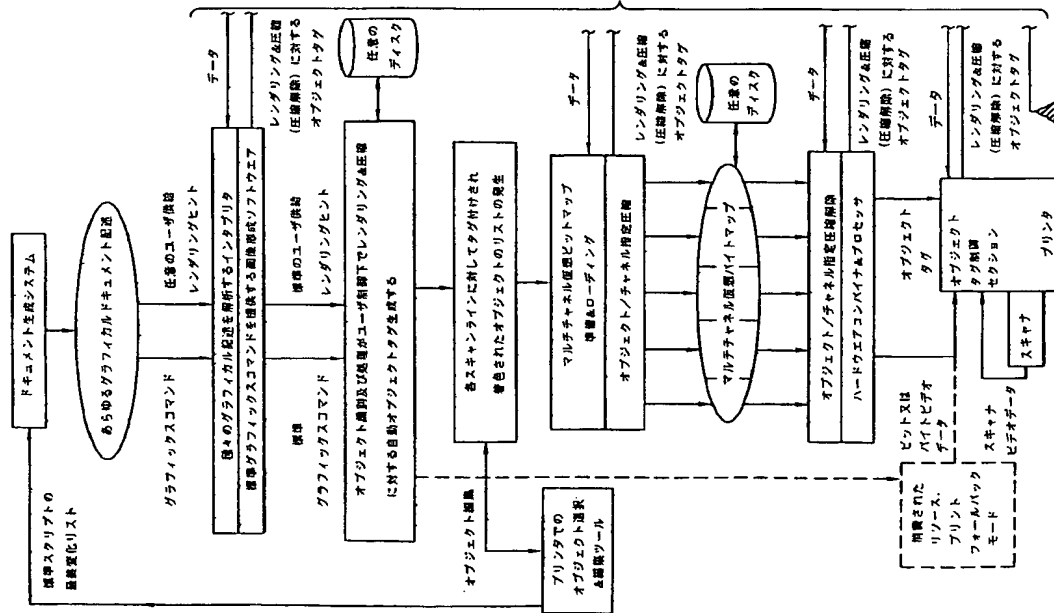
【图31】



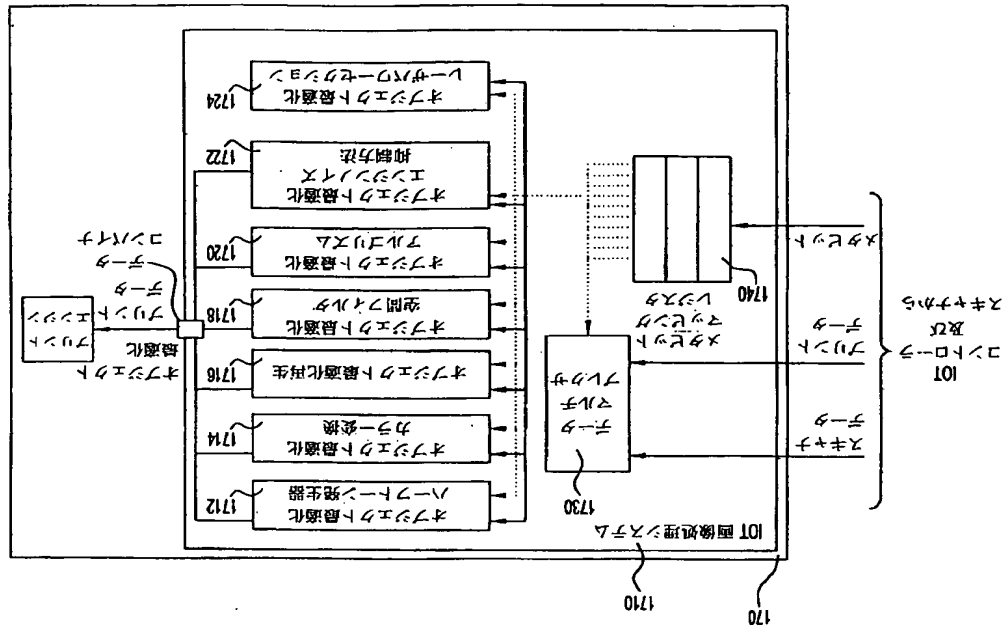
【☒30】



【图32】



【图34】



【図35】

